Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

РОЗРОБКА WEB-РЕСУРСУ ІНТЕРНЕТ-МАГАЗИНУ ЕЛЕКТРОНІКИ

Виконав: студент (4) курсу, групи 3КН–21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Кривовязюк Є.С.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф.КН
Денисюк В.О.
(прізвище та ініціали)

«04» червня 2025 р.

Рецензент: к.т.н, професор каф. АІІТ
Папінов В.М,
(прізвище та ініціали)

«05» червня 2025 р.

Допущено до захисту
Зав. кафедри Яровий А.А.
«06» червня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А. А.
«05» Травня 2025 р

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Кривовязюку Євгенію Сергійовичу

1. Тема роботи: Розробка WEB-ресурсу інтернет-магазину електроніки.
Керівник роботи: Денисюк Валерій Олександрович, к.т.н., доц.
Затверджені наказом вищого навчального закладу "20" Квітня 2025 року №97
2. Термін подання студентом роботи 30 Травня 2025 р.
3. Вихідні дані до роботи: Мова програмування: JavaScript; середовище розробки: Visual Studio Code; програмне забезпечення: Visual Studio Code; Фреймворки: React, Node.js, Express; База даних: MongoDB; Формат обміну даними: JSON; Технології клієнт-серверної взаємодії: REST API; Інтерфейс адміністратора: браузерний (через WEB-панель).
4. Зміст текстової частини (перелік питань, які потрібно розробити): Аналіз предметної області електронної комерції та відомих програм аналогів, постановка задачі для створення веб-ресурсу інтернет-магазину електроніки, проектування системи веб-ресурсу інтернет-магазину електроніки, розробка клієнтської та серверної частини веб-ресурсу інтернет-магазину електроніки, обґрунтування вибору стеку технологій та порівняння альтернатив веб-ресурсу інтернет-магазину електроніки, створення та тестування модулів веб-ресурсу інтернет-магазину електроніки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): Архітектурна схема веб-ресурсу інтернет-магазину, ER-діаграма бази даних веб-ресурсу, схема маршрутів API веб-ресурсу, макет головної сторінки веб-ресурсу інтернет-магазину, макет головної сторінки веб-ресурсу інтернет-магазину, інтерфейс адміністративної панелі для керування товарами та замовленнями, структура клієнтської частини, схема Mongoose-моделі Product (ER-діаграма), структура моделі User, схема роботи маршрутів API.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Денисюк В.О., к.т.н., доц. каф. КН		

7. Дата видачі завдання 05 травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області електронної комерції та відомих програм аналогів	05.05.25	07.05.25	
2	Постановка задачі для створення веб-ресурсу інтернет-магазину електроніки	08.05.25	16.05.25	
3	Проектування системи веб-ресурсу інтернет-магазину електроніки	12.05.25	15.05.25	
4	Розробка клієнтської та серверної частини веб-ресурсу інтернет-магазину електроніки	16.05.25	26.05.25	
5	Обґрунтування вибору стеку технологій та порівняння альтернатив веб-ресурсу інтернет-магазину електроніки	27.05.25	29.05.25	
6	Створення та тестування модулів веб-ресурсу інтернет-магазину електроніки	30.05.25	01.06.25	
7	Оформлення матеріалів до захисту БКР.	02.06.25	04.06.25	

Студент _____ **Кривовязюк Є.С.**
(підпис) *(прізвище та ініціали)*

Керівник роботи _____ **Денисюк В.О.**
(підпис) *(прізвище та ініціали)*

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 79 сторінок формату А4, на яких представлено 17 рисунків, 4 таблиці, список використаних джерел містить 25 найменувань.

Бакалаврська кваліфікаційна робота є частиною комплексного дослідження у галузі веб-програмування та електронної комерції. У межах даної роботи розроблено функціональний веб-ресурс інтернет-магазину електроніки з можливістю перегляду каталогу товарів, додавання до кошика, оформлення замовлення, а також керування контентом через адміністративну панель. Реалізація здійснена із застосуванням технологій JavaScript, React, Node.js, Express, бази даних MongoDB, а також REST API для взаємодії між клієнтською та серверною частинами. Інтерфейс ресурсу є адаптивним, що забезпечує коректне відображення на різних пристроях.

Ключові слова: веб-ресурс, інтернет-магазин, електронна комерція, JavaScript, React, Node.js, REST API, MongoDB.

ABSTRACT

The bachelor's qualification work consists of 79 pages of A4 format, which include 17 figures, 4 tables, and the list of used sources contains 25 items.

The bachelor's qualification work is part of a comprehensive study in the field of web programming and e-commerce. Within the framework of this work, a functional web resource for an online electronics store has been developed with the ability to view the product catalog, add to cart, place an order, and manage content through the administrative panel. The implementation was carried out using JavaScript, React, Node.js, Express technologies, MongoDB databases, and REST API for interaction between the client and server parts. The resource interface is adaptive, which ensures correct display on different devices.

Keywords: web resource, online store, e-commerce, JavaScript, React, Node.js, REST API, MongoDB.

ЗМІСТ

ВСТУП

Актуальність дослідження.

На сьогодні електронна комерція є однією з найдинамічніших галузей цифрової економіки, яка стрімко розвивається в усьому світі. Особливо актуальною вона стала в умовах пандемії, військового часу та загальної цифровізації суспільства. Все більше компаній, підприємств і приватних осіб переходять до онлайн-формату торгівлі, що дає змогу суттєво знизити витрати на ведення бізнесу, розширити цільову аудиторію та автоматизувати процеси замовлення і обслуговування клієнтів.

Інтернет-магазини з продажу електроніки є особливо затребуваними, оскільки попит на техніку стабільно високий і зростає з розвитком технологій. Потенційний клієнт очікує зручного інтерфейсу, швидкого доступу до інформації, можливості фільтрації товарів, здійснення онлайн-замовлення та безпечної оплати. Створення сучасного веб-ресурсу з такими функціями вимагає знань з веб-програмування, проектування інтерфейсів, роботи з базами даних і побудови клієнт-серверних архітектур.

У зв’язку з цим розробка веб-ресурсу інтернет-магазину електроніки є актуальним завданням, яке має як навчальну, так і практичну цінність. Суть дослідження полягає в реалізації повноцінного функціонального сайту з можливістю перегляду товарів, керування замовленнями, реєстрації користувачів та адміністрування контенту.

Метою дослідження і розробки є розширення функціональних можливостей веб-ресурсу інтернет магазину електроніки з базовим функціоналом електронної комерції та адмініструванням, що ґрунтується на сучасних веб-технологіях, за рахунок покращення продуктивності ресурсу.

Об’єктом дослідження є процеси розробки та функціонування веб-застосунків у сфері електронної комерції.

Предметом дослідження є програмні засоби, архітектурні рішення та алгоритми, що реалізують функціональність веб-ресурсу інтернет-магазину.

Задачі дослідження:

- аналіз предметної області електронної комерції та відомих програм аналогів;
- постановка задачі для створення веб-ресурсу інтернет-магазину електроніки;
- проектування системи веб-ресурсу інтернет-магазину електроніки;
- розробка клієнтської та серверної частини веб-ресурсу інтернет-магазину електроніки;
- обґрунтування вибору стеку технологій та порівняння альтернатив веб-ресурсу інтернет-магазину електроніки;
- створення та тестування модулів веб-ресурсу інтернет-магазину електроніки.

Апробація роботи.

Робота пройшла апробацію на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025), 2025р.», доповідь «Розробка WEB-ресурсу інтернет-магазину електроніки» [1].

Публікації.

Електронні тези конференції «Молодь у науці: дослідження, проблеми, перспективи (МН-2025), 2025р.» [1].

Впровадження.

Розроблений програмний продукт може бути впроваджений у навчальні або демонстраційні цілі, а також адаптований для малого бізнесу з продажу електроніки. Архітектура та модульність проекту дозволяють розширювати функціонал і масштабувати ресурс відповідно до вимог конкретного замовника.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ІНТЕРНЕТ-МАГАЗИНУ ЕЛЕКТРОНІКИ

1.1 Аналіз галузі електронної комерції

Електронна комерція наразі є одним із найдинамічніших та найперспективніших секторів цифрової економіки. Вона охоплює широкий спектр бізнес-процесів, пов'язаних з продажем товарів та послуг через Інтернет. Цей формат комерційної діяльності дозволяє здійснювати купівлю та продаж без географічних обмежень, що особливо важливо в умовах глобалізації та постійного розширення ринків збуту.

Дана галузь набула широкого поширення завдяки зростанню кількості користувачів Інтернету, вдосконаленню цифрових технологій та підвищенню рівня довіри до онлайн-платежів. Використання мобільних додатків, персоналізованих інтерфейсів, аналітики поведінки споживачів та автоматизованих систем управління замовленнями стало стандартом у багатьох галузях, що дозволяє компаніям оптимізувати бізнес-процеси, знижувати витрати та покращувати рівень обслуговування.

Особливе місце в електронній комерції займають інтернет-магазини – спеціалізовані веб-ресурси, що забезпечують інтерактивну взаємодію між продавцем та покупцем. Ці ресурси виконують функції каталогізації товарів, обробки замовлень, організації доставки, підтримки зворотного зв'язку та ведення статистики продажів. Висока швидкість обробки транзакцій, можливість інтеграції з платіжними системами та службами доставки, а також цілодобова доступність забезпечили інтернет-магазинам стабільне зростання популярності серед користувачів [3].

На українському ринку електронна комерція також демонструє стабільну позитивну динаміку. За останні роки кількість українських онлайн-платформ значно зросла, а обсяги продажів через Інтернет продовжують зростати, незважаючи на зовнішні виклики, пов'язані з економічною нестабільністю та безпековою ситуацією в країні. Багато підприємств, адаптуючись до нових умов,

активно переводять свою діяльність у цифровий формат, що дозволяє їм залишатися конкурентоспроможними та зберігати клієнтську базу.

Сегмент електроніки займає особливе місце серед інших товарних категорій. Попит на ноутбуки, смартфони, аксесуари, побутову техніку та інші пристрої залишається стабільно високим. Це пояснюється постійним оновленням технологій, бажанням користувачів модернізувати свої пристрої, а також потребами у віддаленій роботі та навчанні. Інтернет-магазини електроніки, на відміну від загальних торгових майданчиків, мають спеціалізовану структуру, орієнтовану на технічну специфікацію товарів, зручну фільтрацію, порівняння характеристик та підтримку супутніх послуг.

З технічного боку сучасні веб-ресурси побудовані за принципом клієнт-серверної архітектури. Клієнтська частина (фронтенд) відповідає за користувацький інтерфейс, візуалізацію даних та взаємодію з сайтом. Вона розробляється з використанням фреймворків JavaScript, таких як React, які дозволяють створювати динамічні односторінкові додатки. Серверна частина (бекенд) обробляє запити, взаємодіє з базою даних, реалізує авторизацію користувачів та іншу бізнес-логіку. Найпоширенішими рішеннями для серверної розробки є платформи Node.js з Express або Python з Django/Flask.

Зберігання даних в інтернет-магазинах здійснюється як у реляційних (MySQL, PostgreSQL), так і в нереляційних базах даних. MongoDB, як приклад документоорієнтованої СУБД, часто обирається для проектів з гнучкою структурою даних, особливо якщо йдеться про продукти з великою кількістю нестандартних характеристик [4].

Окрім технічної сторони, важливо враховувати поведінкові аспекти. Користувачі очікують швидкої навігації, інтуїтивно зрозумілого інтерфейсу, наявності фільтрів, сортування, адаптивного дизайну. UX-дизайн у сфері електронної комерції відіграє вирішальну роль у формуванні враження про сайт та прийнятті рішень про покупку. Наявність відгуків, оцінок товарів, можливість додавання до «обраного» або порівняння значно підвищує ефективність взаємодії [5].

Значення автоматизації також зростає у сфері електронної комерції. Сучасні системи включають модулі для управління запасами, синхронізації з CRM-системами, формування звітів та прогнозування продажів на основі аналітичних даних. Автоматизовані рішення дозволяють скоротити ручну працю, уникнути помилок та підвищити точність обробки замовлень.

Таким чином, електронна комерція – це складна, багаторівнева система, яка включає технічні, економічні та соціальні складові. Розробка власного веб-ресурсу для інтернет-магазину електроніки – актуальне завдання, яке дозволяє поєднати знання в галузі програмування, веб-дизайну, цифрового маркетингу та бізнес-аналізу для створення сучасного конкурентного рішення.

1.2 Аналіз відомих програм аналогів

Сучасний ринок електронної комерції в Україні та світі представлений великою кількістю інтернет-магазинів електроніки, кожен з яких має свою унікальну архітектуру, функціональність та підхід до взаємодії з користувачем. Вивчення їхньої структури, технічних рішень та інтерфейсів дозволяє виявити ключові переваги та недоліки, які необхідно враховувати під час розробки власного веб-ресурсу.

Одним з найпотужніших гравців на українському ринку є Rozetka. Це багатофункціональна платформа електронної комерції, яка пропонує широкий асортимент товарів, включаючи електроніку, побутову техніку, одяг, продукти харчування тощо. Інтерфейс Rozetka інтуїтивно зрозумілий, підтримує адаптивний дизайн, багатомовність, швидку фільтрацію за категоріями, брендами, ціною та рейтингами. Користувачі мають доступ до особистого кабінету, системи замовлень, інтеграції з логістичними службами (зокрема, Новою Поштою) та можливості онлайн-оплати. Rozetka використовує власні CMS-рішення з глибоким налаштуванням, що не підходить для невеликих проектів, але може бути орієнтиром з точки зору функціональності.

Ще один відомий приклад – Comfy, мережа з акцентом на електроніку та технології. Їхній сайт має просту навігацію, акцент на акційні пропозиції,

адаптивну верстку, інтеграцію з мобільним додатком та систему лояльності. Сайт побудований на високопродуктивному бекенді з використанням React та Node.js, а акцент робиться на зручності використання та швидкості обробки замовлень.

MOYO – ще один представник цього сегмента, який вирізняється зручним інтерфейсом для корпоративних клієнтів. Особливість MOYO полягає в його B2B-функціональності: можливість закупівель для компаній, виставлення рахунків-фактур та інтеграція з бухгалтерськими системами. Крім того, MOYO реалізує різні типи фільтрів – від категорій до технічних специфікацій, що корисно при роботі з великою кількістю моделей обладнання.

Серед світових аналогів варто виділити Amazon та Newegg. Amazon – це світовий лідер в електронній комерції, що пропонує тисячі категорій товарів, складну мікросервісну архітектуру, потужну систему рекомендацій на основі ML (машинного навчання), систему зворотного зв'язку та інтеграцію з хмарною інфраструктурою Alexa та AWS. Його досвід може бути корисним у плані UX-дизайну, оптимізації пошуку та логістичних рішень.

Newegg спеціалізується на електроніці, компонентах та програмному забезпеченні. Сайт підтримує розширений пошук, технічні порівняння та глибоку категоризацію, що актуально для технічно підкованої аудиторії. Ця модель найближче відповідає завданням поточного проекту, оскільки орієнтована на споживача, який шукає конкретний пристрій з певними характеристиками.

Серед рішень з відкритим кодом, які можуть слугувати основою для розробки, варто розглянути Magento, OpenCart та PrestaShop. Magento – це платформа електронної комерції з широкими можливостями масштабованості, але вона вимагає високих витрат на хостинг та складна в налаштуванні. OpenCart має спрощений інтерфейс, підтримує теми, плагіни, проста в розгортанні, але обмежена в гнучкості функціональності без доопрацювання. PrestaShop має широку підтримку спільноти, багатомовні можливості та інтеграцію з платіжними системами, але не завжди показує високу продуктивність на великих базах даних.

Таким чином, аналіз існуючих аналогів дозволяє сформулювати ключові вимоги до функціональності, структури та дизайну майбутнього веб-ресурсу. Головними критеріями успіху є швидкість, безпека, інтуїтивно зрозумілий інтерфейс, зручність пошуку товарів, підтримка мобільної версії та інтеграція з логістикою.

1.3 Постановка задачі

Аналіз предметної області, огляд існуючих рішень та вивчення особливостей веб-розробки в сфері електронної комерції дозволяють сформулювати загальну постановку завдання для цієї бакалаврської кваліфікаційної роботи. Головною метою є створення сучасного, адаптивного, функціонального веб-ресурсу для інтернет-магазину електроніки, який відповідає сучасним вимогам до безпеки, зручності та швидкості.

Розробка повинна охоплювати всі етапи життєвого циклу програмного забезпечення: від аналізу вимог та проектування до впровадження, тестування та підготовки до впровадження. В рамках цього процесу необхідно забезпечити логічну структуру клієнтської та серверної частин веб-ресурсу, забезпечити взаємодію між ними через API, а також передбачити системне адміністрування.

Одним з ключових завдань є створення зручного інтерфейсу для кінцевого користувача. Веб-ресурс повинен містити головну сторінку з акцентом на популярні та нові товари, каталог з можливістю фільтрації та сортування за параметрами (категорія, ціна, назва), сторінку окремого товару з повною інформацією, кошик для покупок та функціонал обробки замовлень. Інтерфейс має бути адаптивним, зручним для перегляду як на настільних комп'ютерах, так і на мобільних пристроях.

Ще одним важливим завданням є реалізація серверної частини з підтримкою REST API, яка буде відповідати за обробку запитів від клієнтської частини, авторизацію користувачів, управління продуктами та зберігання даних у базі даних. Особливу увагу слід приділити безпеці – впровадженню токенів

автентифікації, захисту від несанкціонованого доступу до адміністративного функціоналу, шифруванню конфіденційної інформації.

Проект також має передбачати можливість реєстрації користувачів з розділенням ролей: користувач та адміністратор. Користувач повинен мати доступ до каталогу товарів, кошика, оформлення замовлень та перегляду історії покупок. Адміністратор, у свою чергу, повинен мати можливість керувати асортиментом – додавати, редагувати, видаляти товари та керувати категоріями.

До завдань також входять:

1. створення структури бази даних з необхідними зв'язками між сутностями (товари, категорії, користувачі);
2. побудова інтерфейсу для фільтрації та пошуку товарів;
3. реалізація механізму редагування даних про товари через адміністративну панель;
4. створення зручної форми для додавання нових товарів з валідацією;
5. забезпечення збереження стану користувача через JWT-токен;
6. розгортання сервера та запуск клієнтської частини для локального тестування.

Основним завданням цієї роботи є реалізація повноцінного веб-застосунку інтернет-магазину електроніки з можливістю інтеграції в реальне бізнес-середовище. Отримане рішення має бути, безпечним, простим у використанні та легко підтримуватися в майбутньому.

1.4 Висновок до розділу 1

У ході першого розділу було проведено ґрунтовний аналіз індустрії електронної комерції, розглянуто сучасні вимоги до веб-ресурсів інтернет-магазинів, а також сформульовано технічні та функціональні завдання, які необхідно реалізувати в рамках бакалаврської кваліфікаційної роботи.

Загальний огляд розвитку ринку електронної комерції в Україні та світі показав його стабільне зростання, що зумовлене активною цифровізацією суспільства, розвитком платіжних інструментів, логістики та мобільного

Інтернету. У зв'язку з цим створення інтернет-магазину електроніки є не лише актуальним завданням, але й важливим кроком до адаптації бізнесу до потреб сучасного користувача. Було виявлено, що ключовими факторами успіху в галузі є швидкість роботи сайту, адаптивність інтерфейсу, безпека персональних даних та зручність навігації.

У підрозділі 1.2 було проведено аналіз функціональних можливостей популярних веб-ресурсів, що працюють в електронній комерції. Розглянуто сильні та слабкі сторони таких систем, як Rozetka, Comfy, MOYO, Eldorado та інші. Аналіз показав, що хоча ці сервіси надають широкий функціонал, їхня складна архітектура та закриті системи не дозволяють гнучко адаптувати рішення до потреб малого бізнесу. Це відкриває перспективи для розробки окремих веб-ресурсів з відкритим кодом, які можна модифікувати та розширювати залежно від вимог проекту.

У підрозділі 1.3 чітко окреслено постановку завдання на основі проаналізованих вихідних даних. Були сформульовані вимоги до архітектури клієнтської та серверної частин майбутнього застосунку, а також визначено перелік основних функцій, які мають бути реалізовані. Зокрема, акцент було зроблено на реалізації пошуку та фільтрації товарів, створенні адміністративної панелі для управління контентом, інтеграції з базою даних, забезпеченні реєстрації та авторизації користувачів, а також дотриманні сучасних стандартів безпеки та UX/UI дизайну [6].

Підсумовуючи, можна зазначити, що аналіз підтвердив практичну доцільність впровадження веб-ресурсу для інтернет-магазину електроніки. Були визначені ключові модулі системи, встановлені пріоритети функціональності та обґрунтовано вибір технологічного підходу. В результаті цього етапу сформувалося чітке розуміння подальших кроків розробки, яке буде вкладено в основу наступних розділів.

2. РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ WEB-РЕСУРСУ ІНТЕРНЕТ-МАГАЗИНУ ЕЛЕКТРОНІКИ

2.1 Проектування системи

Розробка веб-ресурсу інтернет-магазину електроніки потребує ретельного проектування як технічної архітектури, так і структури даних та інтерфейсу. На цьому етапі особливу увагу було приділено вибору відповідних технологій, моделюванню логіки взаємодії користувача із системою, а також практичній реалізації функціоналу, що відповідає сформульованим вимогам [7-8].

Архітектурною основою розробленої системи стала класична модель клієнт-серверної взаємодії (рис. 2.1). Клієнтська частина реалізована на основі бібліотеки React, що дозволяє створити інтерактивний інтерфейс з використанням компонентного підходу [9]. Серверна логіка побудована за допомогою платформи Node.js та фреймворку Express [10]. Ці компоненти взаємодіють між собою за допомогою REST API, що забезпечує структуровану передачу даних у форматі JSON. Всі основні дані зберігаються в базі даних MongoDB, яка дає змогу працювати з гнучкими документно-орієнтованими структурами.

Для забезпечення безпеки користувацьких даних застосовано бібліотеки JWT для створення токенів доступу та bcrypt для хешування паролів. При реєстрації паролі не зберігаються у відкритому вигляді, а хешуються перед записом у базу даних. Аналогічно, при авторизації введений користувачем пароль порівнюється з хешем у базі за допомогою функції compare [11].

Окрім базового функціоналу, система передбачає розділення прав доступу для різних ролей – адміністратора та звичайного користувача. Адміністративна панель дозволяє керувати товарами, замовленнями та переглядати активність користувачів. Клієнтський інтерфейс адаптований до мобільних пристроїв, що підвищує зручність використання ресурсу.

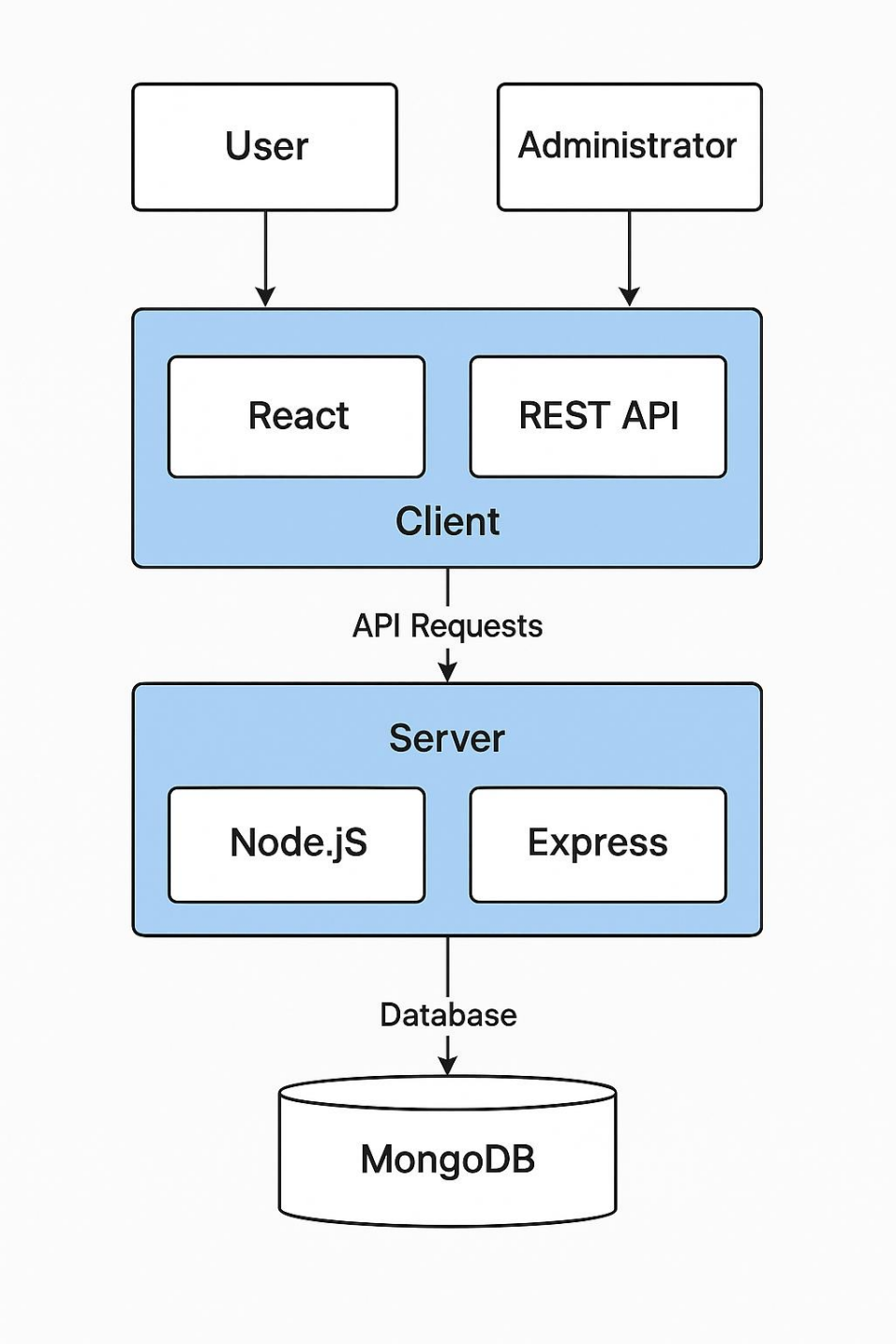

Рисунок 2.1 – Архітектурна схема веб-ресурсу інтернет-магазину

Інформційна частина системи організована у вигляді бази даних, яка включає кілька ключових сутностей: товари, користувачі та замовлення (рис. 2.2). Колекція товарів містить відомості про назву, опис, ціну, категорію та ілюстрації кожної позиції. Дані користувачів включають облікову інформацію, роль у системі та захищені паролі. Структура замовлень передбачає збереження

переліку обраних товарів, контактної інформації замовника та поточного статусу виконання. Для забезпечення коректної роботи та валідації даних реалізовано перевірки як на рівні моделі, так і безпосередньо при надходженні запитів до сервера. Нижче представлено приклад схеми колекції "Products":

{ "name": "Smartphone Xiaomi Redmi 10", "price": 5999, "description": "Бюджетний смартфон з екраном 6.5", FHD+, 50 МП камера", "category": "Смартфони", "imageUrl": "https://example.com/redmi10.jpg" }

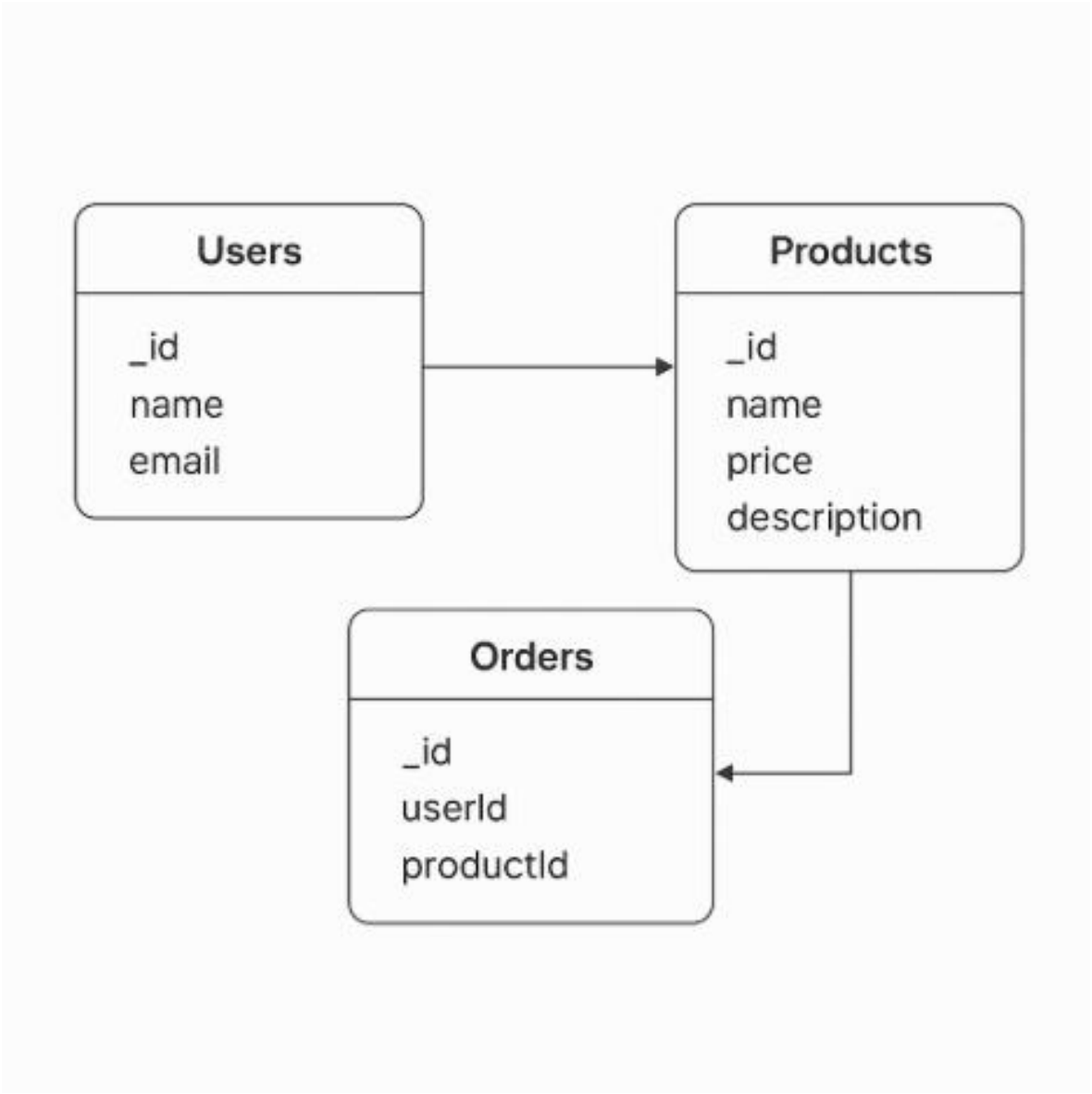

Рисунок 2.2 – ER-діаграма бази даних веб-ресурсу

У логіці взаємодії клієнта із сервером застосовано бібліотеку Axios, що дає змогу надсилати асинхронні запити до бекенду та отримувати відповіді без перезавантаження сторінки [12]. Наприклад, запит на отримання списку товарів реалізовано наступним чином:

axios.get('http://localhost:5000/api/products').then(res => setProducts(res.data));

На стороні сервера всі запити обробляються Express. Роутинг структуровано за логічними напрямками: обробка товарів, користувачів та замовлень винесена у відповідні модулі (рис. 2.3). Для прикладу, створення нового товару обробляється наступним кодом:

app.post('/api/products', async (req, res) => {const { name, price, description, category} = req.body;

const newProduct = new Product({ name, price, description, category });

await newProduct.save();

res.status(201).json(newProduct);

});

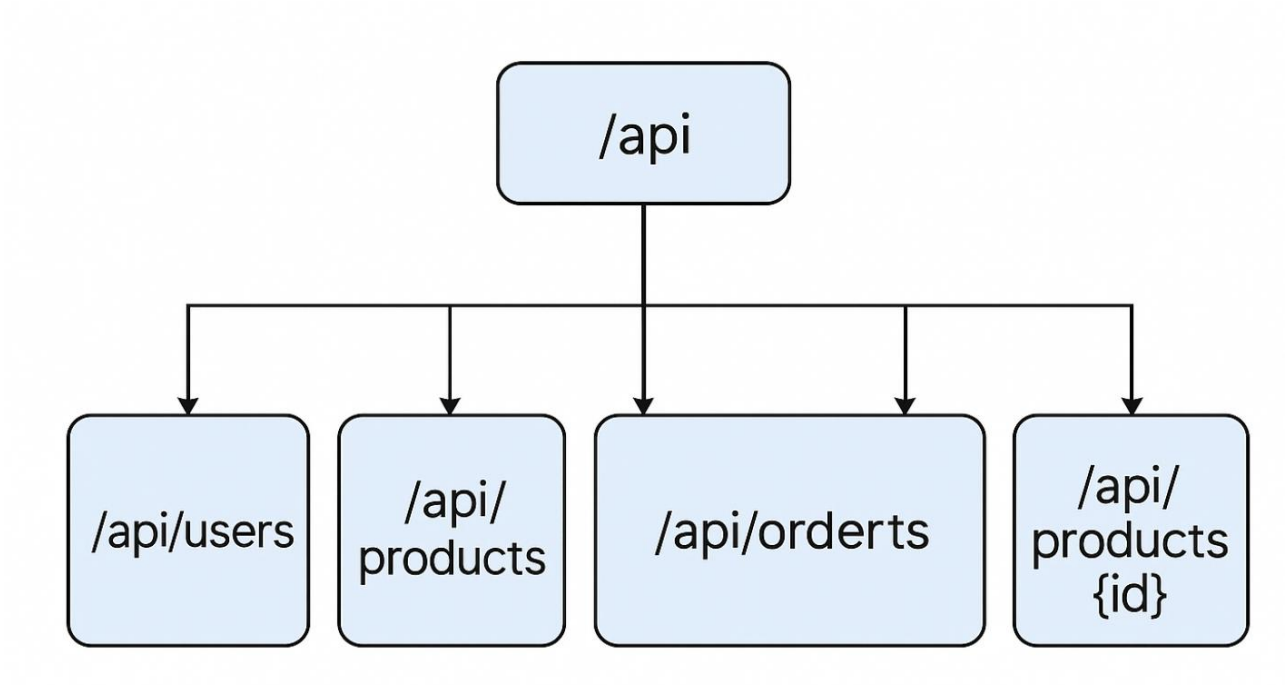

Рисунок 2.3 – Схема маршрутів API веб-ресурсу

Окремої уваги заслуговує система ролей користувачів. У межах проекту передбачено поділ на дві основні ролі – звичайний користувач та адміністратор. Залежно від ролі, користувач отримує доступ до певного функціоналу: звичайний користувач має змогу переглядати товари, додавати їх до кошика та оформляти замовлення, тоді як адміністратор має розширені повноваження на додавання, редагування та видалення товарів, перегляд замовлень та керування статусами.

Клієнтська частина розроблена з урахуванням сучасних стандартів фронтенд-розробки (рис. 2.4). React дозволяє використовувати функціональні компоненти з хуками для управління станом, що підвищує гнучкість та підтримуваність коду. Усі компоненти згруповано за функціональністю: наприклад, компоненти каталогу, товарної картки, кошика, авторизації та кабінету користувача. Використання бібліотеки TailwindCSS дало змогу прискорити процес стилізації інтерфейсу та забезпечити адаптивність без потреби створювати окремі медіазапити вручну. Основні маршрути додатку реалізовані через бібліотеку react-router [13].

Інтерфейс користувача розроблено таким чином, щоб він був зрозумілим, адаптивним та легким у навігації. Головна сторінка відображає доступний асортимент продукції, кожен товар має окрему сторінку з детальним описом. Реалізовано функціонал кошика, який дозволяє додавати та вилучати товари, переглядати підсумкову суму замовлення. Крім того, користувач має можливість зареєструватися або увійти в особистий кабінет, де можна відстежувати історію замовлень.

Для зручності взаємодії було реалізовано повідомлення про успішні дії, помилки авторизації та підказки при заповненні форм. У разі спроби доступу до заборонених сторінок користувача автоматично перенаправляє на головну або на сторінку входу. Завдяки SPA-підходу (Single Page Application) усі переходи між сторінками відбуваються без перезавантаження, що покращує загальний користувацький досвід.

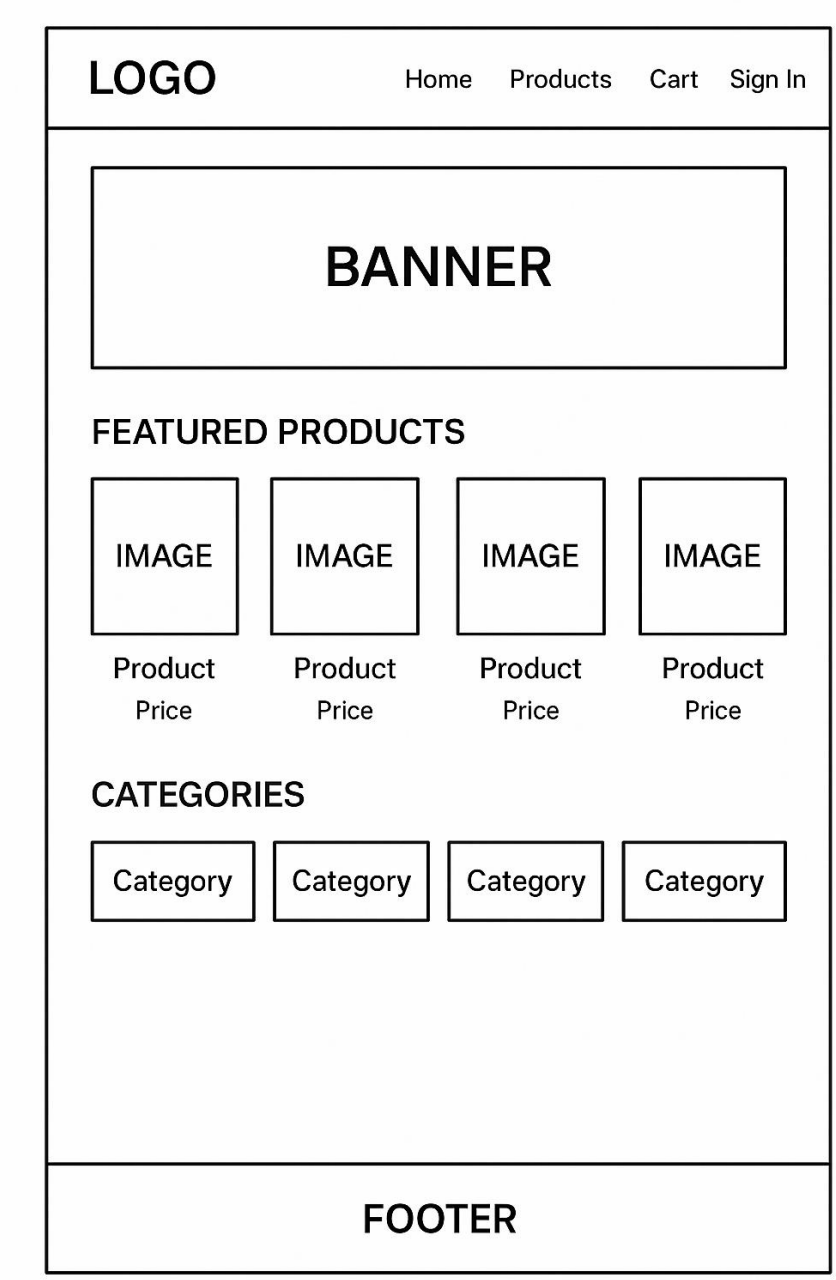

Рисунок 2.4 – Макет головної сторінки веб-ресурсу інтернет-магазину

Особливу увагу приділено адміністративній частині системи (рис. 2.5). Вона доступна лише авторизованим користувачам із правами адміністратора і дозволяє переглядати повний список товарів, редагувати їх характеристики, додавати нові позиції до каталогу, а також опрацьовувати замовлення та змінювати їх статус. Усі дії супроводжуються відповідним візуальним інтерфейсом, що не потребує технічних знань для керування контентом.

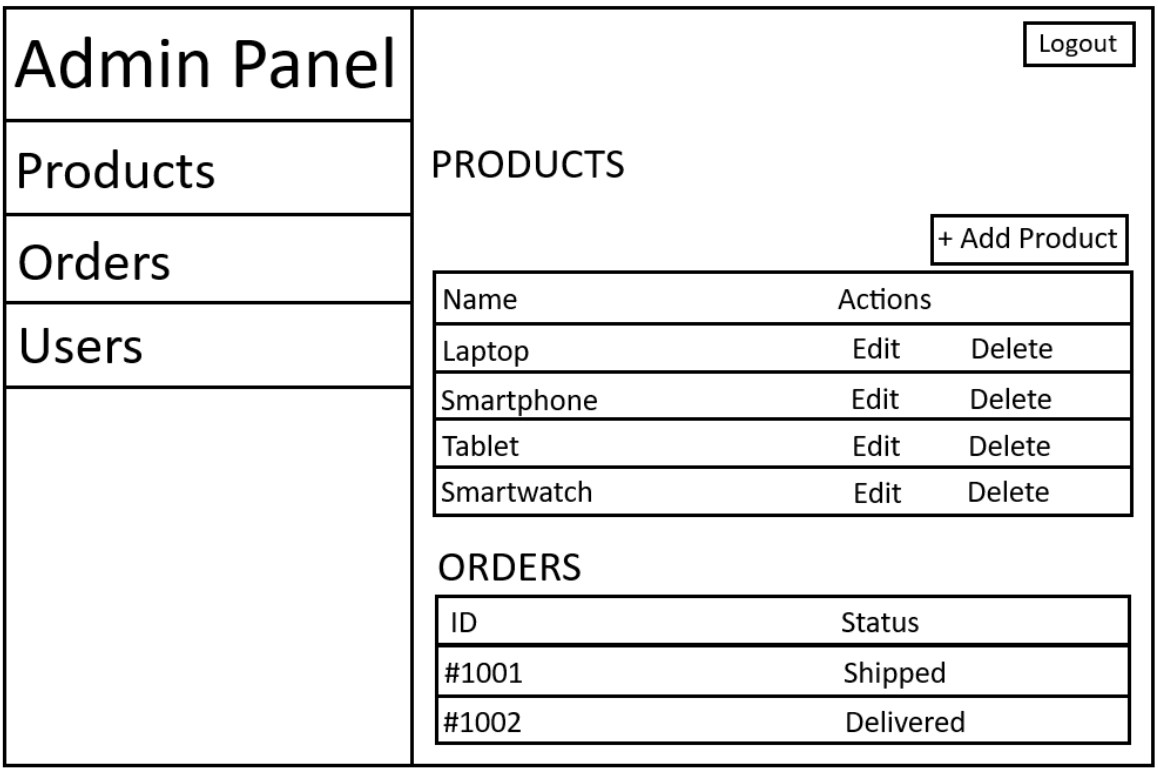

Рисунок 2.5 – Інтерфейс адміністративної панелі для керування товарами та замовленнями

Під час реалізації системи також було проведено базове тестування функціоналу. Кожен модуль перевірявся на наявність типових помилок: відсутність даних у формі, некоректний запит до API, спроба доступу до захищених маршрутів без авторизації. Усі знайдені недоліки було усунуто. Для тестування використовувались як інтеграційні запити, так і поведінкові сценарії з боку користувача.

Окремим етапом стало створення документації до системи. Вона містить інструкції щодо запуску клієнтської та серверної частин, підключення до бази даних, приклади запитів до API та типові сценарії використання. Така документація стане в нагоді як для подальших розробників, так і для адміністраторів ресурсу при впровадженні проекту.

Інтерфейс веб-додатку розроблено з урахуванням вимог адаптивності. Це означає, що сайт коректно відображається як на широкоформатних моніторах, так і на екранах мобільних пристроїв. Реалізовано базові принципи доступності:

достатній контраст тексту, зрозумілі підписи для форм, можливість навігації з клавіатури. Це підвищує зручність користування для ширшої аудиторії.

Окремо варто розглянути перспективи розвитку системи. Зокрема, вже на етапі проектування було передбачено можливість масштабування та інтеграції з платіжними сервісами (LiqPay, PayPal, Stripe), службами доставки (Нова Пошта, Meest Express), а також системами аналітики (Google Analytics, власні дашборди). Така модульна побудова дозволяє адаптувати веб-ресурс до потреб конкретного бізнесу з мінімальними зусиллями.

У результаті реалізації отримано прототип веб-ресурсу інтернет-магазину електроніки, який охоплює повний життєвий цикл взаємодії користувача із системою: від ознайомлення з товаром до оформлення покупки. Також створено окрему панель керування контентом для адміністратора. Розроблена структура коду дозволяє легко масштабувати систему та доповнювати її новими функціональними модулями без значних змін до існуючої архітектури. Такий підхід відкриває перспективи для подальшого розвитку проекту – зокрема, інтеграції платіжних систем, розширення функціоналу особистого кабінету, впровадження аналітики продажів, а також адаптації до мобільних додатків.

Таким чином, реалізована система відповідає поставленим завданням, дозволяє вирішувати актуальні потреби користувача і є придатною для розгортання у реальному середовищі або для подальшого вдосконалення в рамках комерційного або навчального проекту.

З метою розширення функціональності системи розглянуто можливість впровадження додаткових модулів, які можуть бути реалізовані у наступних ітераціях розробки. Одним із таких напрямів є впровадження системи рекомендацій на основі історії переглядів та покупок користувачів. Для реалізації такого механізму доцільним є використання алгоритмів машинного навчання або простіших евристичних підходів, які враховують популярність товарів, частоту їх додавання до кошика та співвідношення переглядів до замовлень.

Ще одним важливим напрямом розвитку є реалізація функціоналу відстеження замовлень. Це дозволить користувачам у режимі реального часу бачити статус їхнього замовлення, а адміністраторам – своєчасно оновлювати дані про доставку. Така функціональність може бути реалізована через окремий маршрут у REST API з фільтрацією за ідентифікатором замовлення та розмежуванням доступу до даних.

Крім того, у системі передбачено можливість реалізації відгуків на товари. Кожен зареєстрований користувач зможе залишити текстовий коментар після покупки, що підвищить рівень довіри до сайту та дозволить іншим користувачам приймати більш обґрунтовані рішення. У майбутньому можливе також додавання рейтингової системи, яка базуватиметься на оцінках від 1 до 5 балів. Це потребуватиме створення додаткової колекції в базі даних, пов’язаної з товарами та користувачами.

З технічного боку доцільним буде впровадження кешування даних для зменшення навантаження на сервер. Наприклад, часто запитувані товари або результати фільтрації можна тимчасово зберігати у кеші, що зменшить кількість звернень до бази даних і покращить швидкодію системи. Для цього можна використовувати бібліотеки на кшталт Redis.

Ще одним напрямом удосконалення є створення мобільної версії або PWA (Progressive Web Application), що дозволить користувачам працювати з сайтом як із класичним мобільним застосунком. PWA забезпечує кешування сторінок, офлайн-режим, сповіщення та іконку на головному екрані пристрою. Це суттєво розширить аудиторію користувачів і підвищить конкурентоспроможність продукту.

Варто також відзначити перспективи інтеграції CRM-системи для обліку клієнтів, генерації звітів і комунікації з покупцями. Наприклад, після оформлення замовлення система може автоматично надсилати підтвердження, подяку або повідомлення про акції. Це сприятиме формуванню лояльної клієнтської бази та підвищенню рівня продажів.

У контексті адміністрування ресурс може бути доповнений журналом дій (логом), що фіксуватиме всі зміни в системі: додавання або видалення товарів, зміну цін, оновлення статусів замовлень. Це дозволить відстежувати дії персоналу і в разі потреби – виявляти помилки або зловживання. Також така система є важливою для аудитів і безпеки загалом.

З точки зору дизайну інтерфейсу, можливим покращенням є реалізація темної теми, яка зменшує навантаження на очі користувача при використанні сайту вночі. Також зручним буде додавання мовної підтримки, що дозволить обрати мову інтерфейсу та орієнтуватись на ширшу аудиторію – зокрема, на клієнтів з-за кордону або людей з іноземною мовою спілкування.

Додатковим плюсом стане реалізація модуля порівняння товарів. Це дасть змогу користувачам обрати декілька позицій і порівняти їх за ключовими характеристиками. Така функціональність є зручною для технічно складних товарів, як-от ноутбуки, смартфони чи побутова техніка.

Для реалізації вищезазначених можливостей передбачається масштабування структури проекту, а також розширення бази даних новими колекціями або властивостями. З цією метою архітектура була спроектована з урахуванням можливості подальшого доповнення – код написано у вигляді окремих модулів, що легко тестуються і змінюються незалежно один від одного.

У підсумку можна зазначити, що вже реалізована система є лише базовим етапом повноцінного рішення для електронної торгівлі. Вона створює фундамент для впровадження нових функцій і масштабування, з урахуванням потреб конкретного бізнесу або стартапу. Гнучкість архітектури, використання сучасного технологічного стеку та підтримка принципів модульності роблять її придатною як для локального використання, так і для розгортання на широке коло клієнтів у майбутньому.

З метою покращення загального користувацького досвіду (UX), доцільно передбачити реалізацію персоналізованих елементів інтерфейсу, які адаптуватимуть вміст під конкретного користувача. Наприклад, на головній

сторінці може відображатися історія останніх переглядів або персональні рекомендації товарів. Такий підхід підвищує залученість користувача та сприяє збільшенню конверсії.

У майбутньому до системи може бути інтегрований механізм обліку знижок і промокодів. Це дозволить адміністраторам формувати акційні пропозиції, а користувачам – отримувати знижки при введенні відповідного коду під час оформлення замовлення. Для реалізації такого функціоналу слід створити додаткову сутність у базі даних, що міститиме параметри дії знижки: назву, відсоткове значення, термін дії та умови активації.

Також актуальним є створення модулю сповіщень. Це можуть бути як email-повідомлення, так і push-сповіщення для браузера. Завдяки такому інструменту користувач буде вчасно інформований про нові товари, акції або зміни у статусі свого замовлення. Впровадження такого модуля підвищує довіру до платформи та покращує взаємодію з клієнтом.

Не менш важливим є забезпечення SEO-оптимізації веб-ресурсу. Для підвищення видимості сайту у пошукових системах необхідно реалізувати статичні URL-адреси для кожного товару, використовувати мета-теги для описів сторінок, а також оптимізувати структуру HTML-документів. Це дозволить ресурсу з’являтися вище у результатах пошуку Google, що вкрай важливо для реального комерційного використання.

Щоб забезпечити контроль продуктивності, рекомендується впровадити інструменти моніторингу, зокрема Google Lighthouse, який дозволяє аналізувати час завантаження, інтерактивність та інші показники продуктивності. Також можливе використання сторонніх сервісів на кшталт Sentry або LogRocket для збору інформації про помилки та дії користувача.

У процесі розробки проводилась перевірка роботи веб-ресурсу в різних браузерах та на різних пристроях, що дало змогу виявити та усунути проблеми з відображенням певних компонентів. Для цього використовувались інструменти

розробника в браузерах Chrome, Firefox та Safari. Завдяки адаптивному дизайну сайт виглядає коректно як на десктопах, так і на мобільних пристроях.

2.2 Розробка клієнтської частини

Клієнтська частина веб-ресурсу інтернет-магазину електроніки була реалізована з використанням JavaScript-бібліотеки React, яка є популярним інструментом для створення сучасних односторінкових веб-додатків (SPA). Завдяки компонентній архітектурі React дозволяє розділити інтерфейс на незалежні частини, кожна з яких реалізує власну логіку відображення та взаємодії з користувачем [14].

Основним завданням клієнтської частини є формування зручного, адаптивного та функціонального інтерфейсу для взаємодії користувача з сервісом. У процесі реалізації застосовувалися такі технології, як React Router для маршрутизації сторінок, Axios для відправлення HTTP-запитів, а також Tailwind CSS для адаптивної стилізації компонентів [15].

Структура клієнтської частини (рис. 2.6) була реалізована у вигляді директорій з компонентами, сторінками та службовими модулями. Основними директоріями є:

1) components/ – окремі модулі, що відповідають за окремі частини інтерфейсу (форма, список товарів, фільтр, навігація);
2) pages/ – сторінки для рендерингу в рамках маршрутизатора;
3) services/ – логіка взаємодії з API (Axios-запити);
4) App.jsx – головний компонент, який об’єднує всі сторінки та реалізує маршрутизацію;
5) index.js – точка входу в додаток, що впроваджує React-компоненти у DOM-дерево.

Головна сторінка додатку (HomePage) містить перелік товарів з можливістю пошуку та фільтрації за категорією та кнопками для перегляду, редагування та видалення (у разі адміністративного доступу) (рис. 2.7).

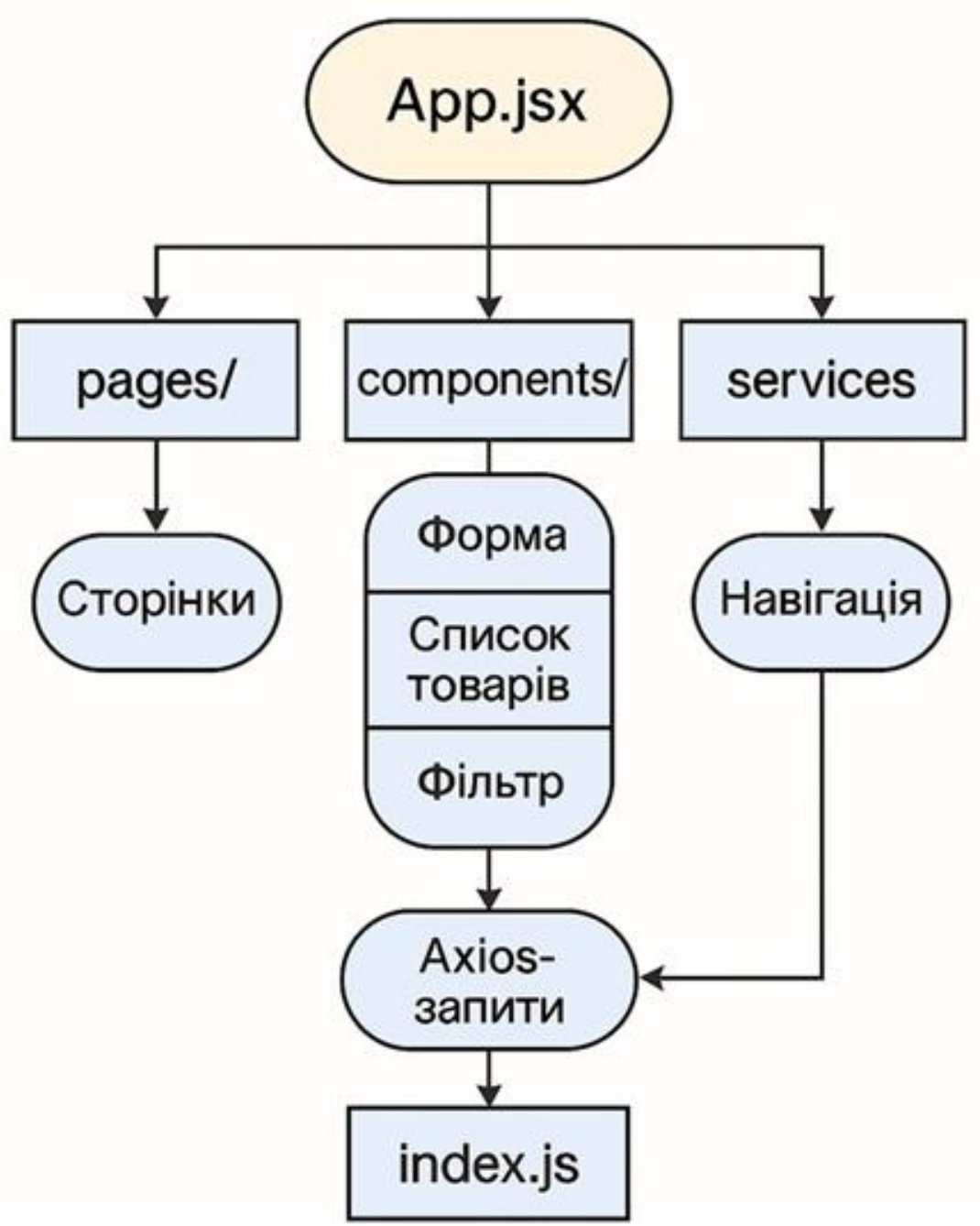

Рисунок 2.6 – Структура клієнтської частини

Окремим компонентом реалізовано форму AddProductForm, яка дозволяє вводити назву, опис, ціну, зображення та категорію товару (рис. 2.8). У разі редагування дані попередньо підставляються з відповідного товару, що забезпечує зручність для адміністратора.

Валідація полів здійснюється як на клієнтському, так і на серверному боці. Після успішної відправки товар автоматично додається у список.

В залежності від ролі користувача (user або admin) динамічно змінюється інтерфейс та доступні функції.

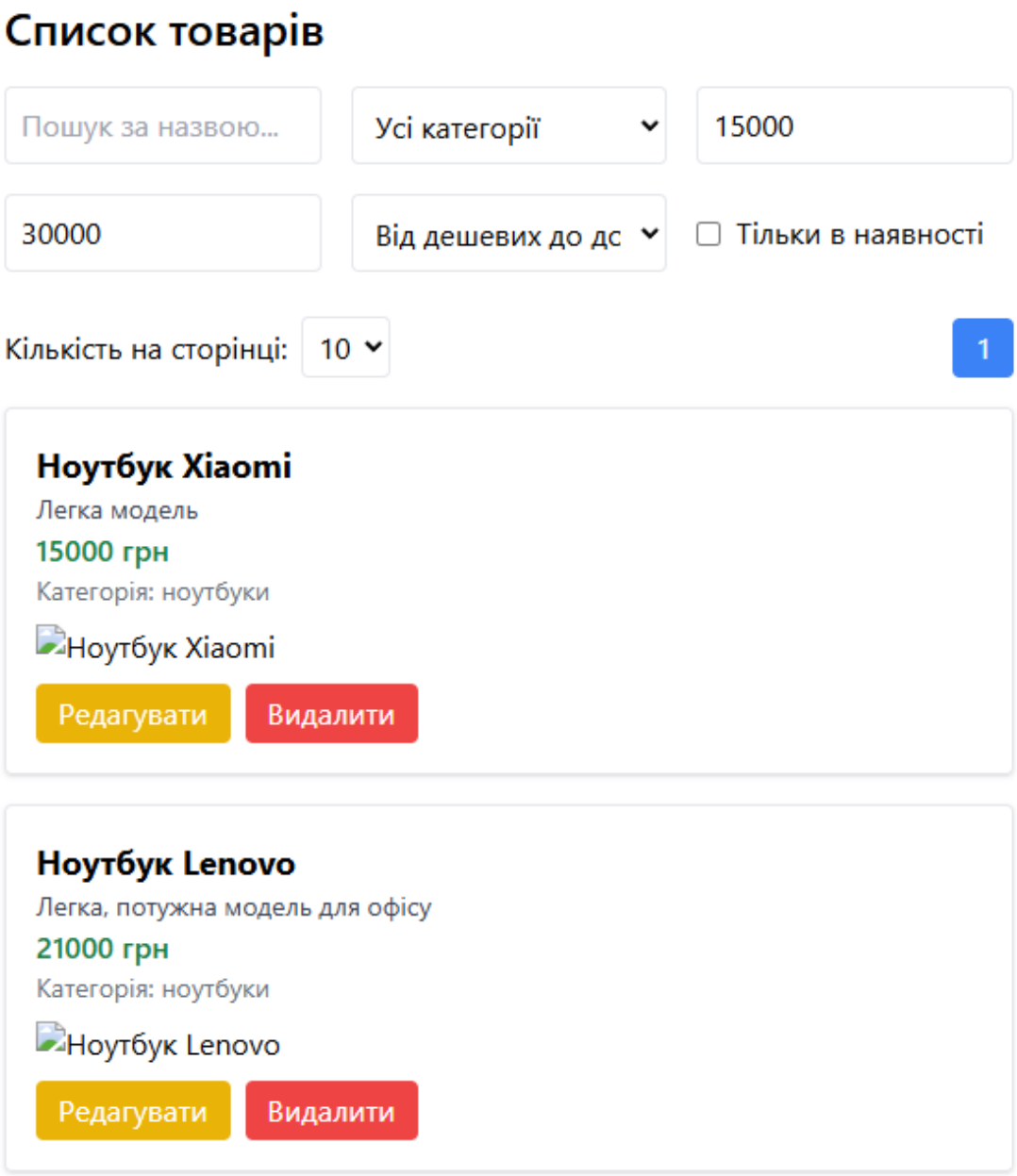

Рисунок 2.7 – Відображення списку товарів з фільтрами та кнопками для перегляду/редагування/видалення

Для взаємодії з авторизаційною системою було реалізовано дві форми – для реєстрації нового користувача та для входу в акаунт. Після входу користувач отримує токен, який зберігається в локальному сховищі браузера (localStorage) і автоматично додається до кожного запиту, що потребує авторизації.

Tailwind CSS дозволяє створювати адаптивну верстку без складного перевантаження CSS-файлів. Усі компоненти відображаються коректно на мобільних пристроях, планшетах і настільних екранах.

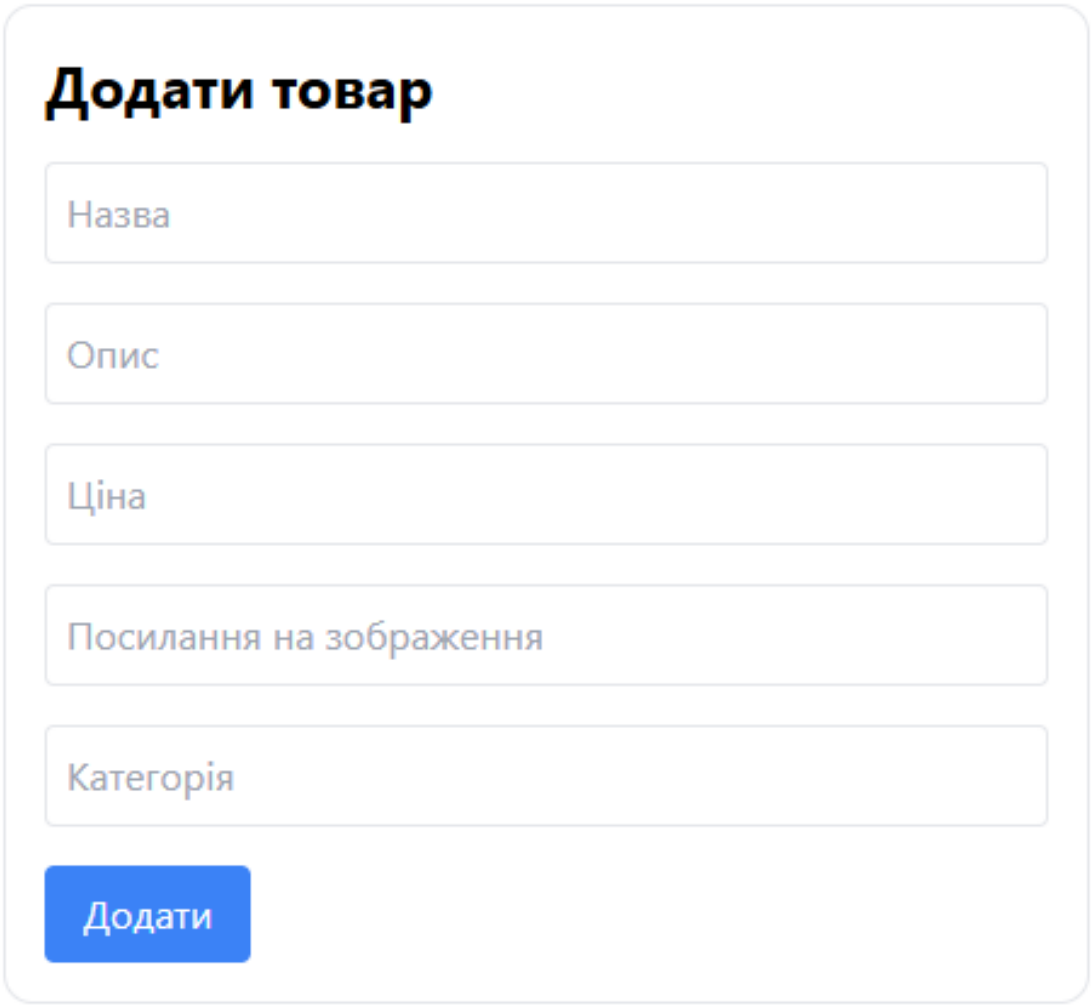

Рисунок 2.8 – Форма додавання нового товару

У додатку реалізовано сповіщення для користувача про успішні або помилкові дії (додавання товару, редагування, видалення, помилки форми). Для цього використано React-хуки (useState, useEffect) та умовне рендерення блоків повідомлень.

JWT-токен, отриманий після входу, передається в заголовку запитів (Authorization: Bearer ...) для захисту ресурсів. У разі завершення сесії або

недійсного токена користувач автоматично перенаправляється на сторінку авторизації.

У додатку реалізовано особистий кабінет користувача, в якому зареєстрований користувач може переглядати свої дані, редагувати профіль (змінювати email, пароль), а також переглядати історію замовлень. Користувач також має можливість видалити свій обліковий запис. Для захисту доступу до кабінету використано перевірку токену авторизації та валідацію ролі користувача.

Важливою функціональністю клієнтської частини є кошик користувача, який реалізовано через глобальний стан (наприклад, Context API або Redux). Користувач може додавати товари до кошика, змінювати кількість одиниць, видаляти позиції. Після формування списку замовлення, натискання кнопки "Оформити" ініціює процес оформлення замовлення, який включає вибір способу оплати, адреси доставки та підтвердження покупки.

Окрема частина клієнтського інтерфейсу – це адміністративна панель, яка доступна тільки для користувачів з роллю admin. Через цю панель адміністратор може додавати, редагувати та видаляти товари, переглядати список користувачів, змінювати їх статус або блокувати. Також адмінісртативна панель дозволяє переглядати всі замовлення, змінювати їх статус та аналізувати статистику продажів за період.

2.3 Розробка серверної частини

Серверна частина інтернет-магазину реалізована за допомогою JavaScript-фреймворку Express.js, який працює поверх середовища виконання Node.js [16]. Система побудована за принципами REST-архітектури, що забезпечує простоту масштабування, логічну структуру взаємодії з клієнтською частиною та підтримку сучасних веб-технологій.

Для забезпечення коректної обробки HTTP-запитів до серверної частини було підключено основні модулі проміжного програмного забезпечення, зокрема express.json() для парсингу вхідних JSON-даних, а також cors — для обробки

міждоменної взаємодії між клієнтом та сервером. Це дозволяє ефективно налагоджувати взаємодію між окремими частинами веб-застосунку, що працюють на різних портах.

Файлова структура серверної частини організована за принципами модульності. Окремий каталог routes/ містить маршрути для роботи з користувачами, продуктами, замовленнями та автентифікацією. Контролери (controllers/) містять логіку обробки запитів, уникаючи дублювання коду та покращуючи читабельність.

Особливу увагу було приділено реалізації безпеки: система використовує бібліотеку jsonwebtoken для створення токенів автентифікації. Це дозволяє зберігати сеанс користувача в безпечному форматі та обмежувати доступ до захищених маршрутів відповідно до ролі користувача — адміністратор або звичайний користувач.

REST API побудовано за принципом CRUD (Create, Read, Update, Delete), що забезпечує повну взаємодію з продуктами, категоріями, користувачами та замовленнями. Кожен маршрут відповідає окремому ресурсу та реалізується через відповідні HTTP-методи — GET, POST, PUT, DELETE.

Після ініціалізації проекту сервер конфігурується для використання порту (5000), з підключенням до бази даних MongoDB. Для цього використано бібліотеку mongoose, яка дозволяє працювати з документно-орієнтованою моделлю зручним об’єктно-орієнтованим інтерфейсом [17].

При роботі з базою даних у серверній частині створено модель Product, яка визначає структуру об’єкта товару в базі даних: назва, опис, ціна, зображення, категорія, наявність продукту (рис. 2.9). Збереження та обробка даних виконується з використанням CRUD-операцій (створення, читання, оновлення, видалення).

На діаграмі показано, що кожен товар (Product) містить посилання на відповідну категорію у вигляді поля Category. Це дозволяє реалізувати фільтрацію, угруповання та тематичне сортування товарів у межах інтернет-магазину.

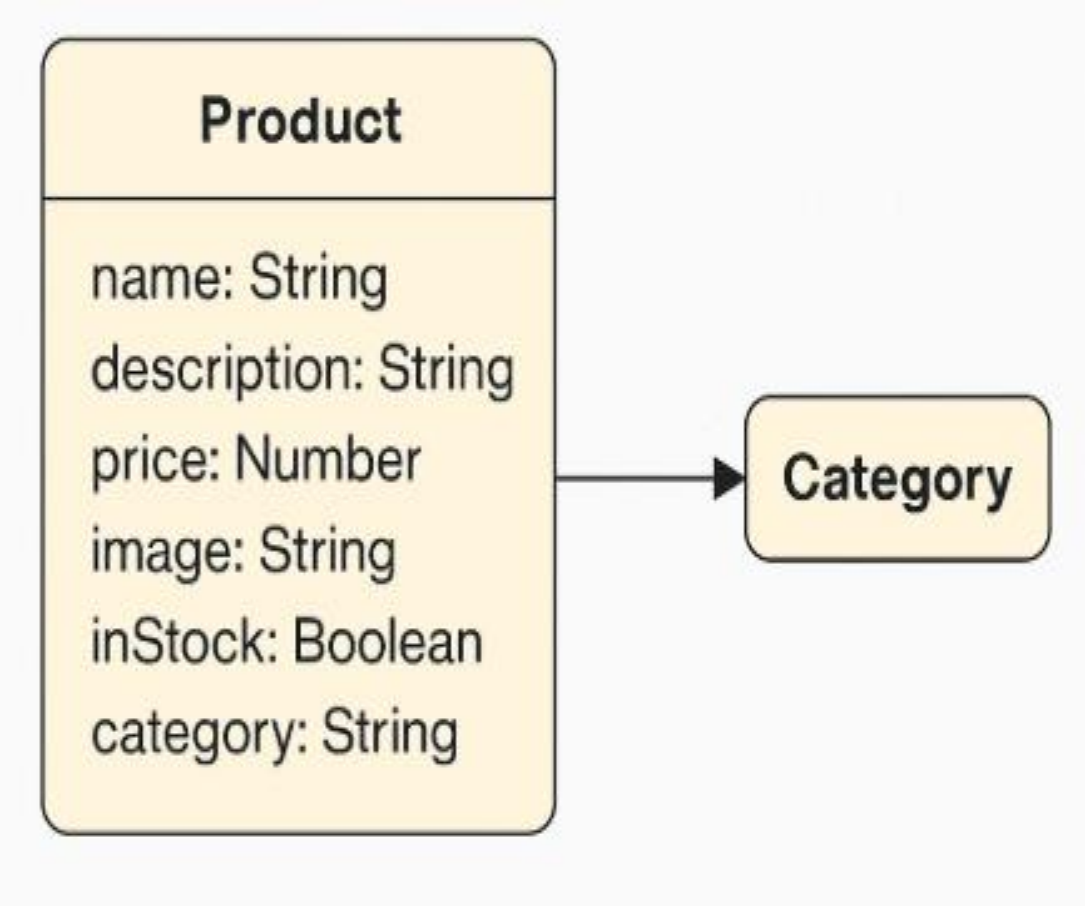

Рисунок 2.9 – Схема Mongoose-моделі Product (ER-діаграма)

Аналогічним чином реалізовано модель User, яка містить інформацію про користувачів: email, хешований пароль, роль (user або admin) та дату реєстрації (рис. 2.10). Паролі зберігаються у захищеному вигляді завдяки бібліотеці bcrypt.

Для безпеки взаємодії користувача з системою реалізовано реєстрацію з автоматичним хешуванням пароля та авторизацію на основі JWT-токенів (JSON Web Token). Токени генеруються при успішному вході та зберігаються на клієнті. При кожному запиті до серверу, який вимагає авторизації, клієнт надсилає токен у заголовку [18].

Для захисту серверних маршрутів (рис. 2.11) створено middleware, який перевіряє валідність токена та, при необхідності, роль користувача. Таким чином, деякі маршрути доступні лише адміністраторам, а інші – тільки зареєстрованим користувачам, які мають роль user або admin. Відповідно до ролі користувача також відображаються доступні функції веб-ресурсу на сторінці.

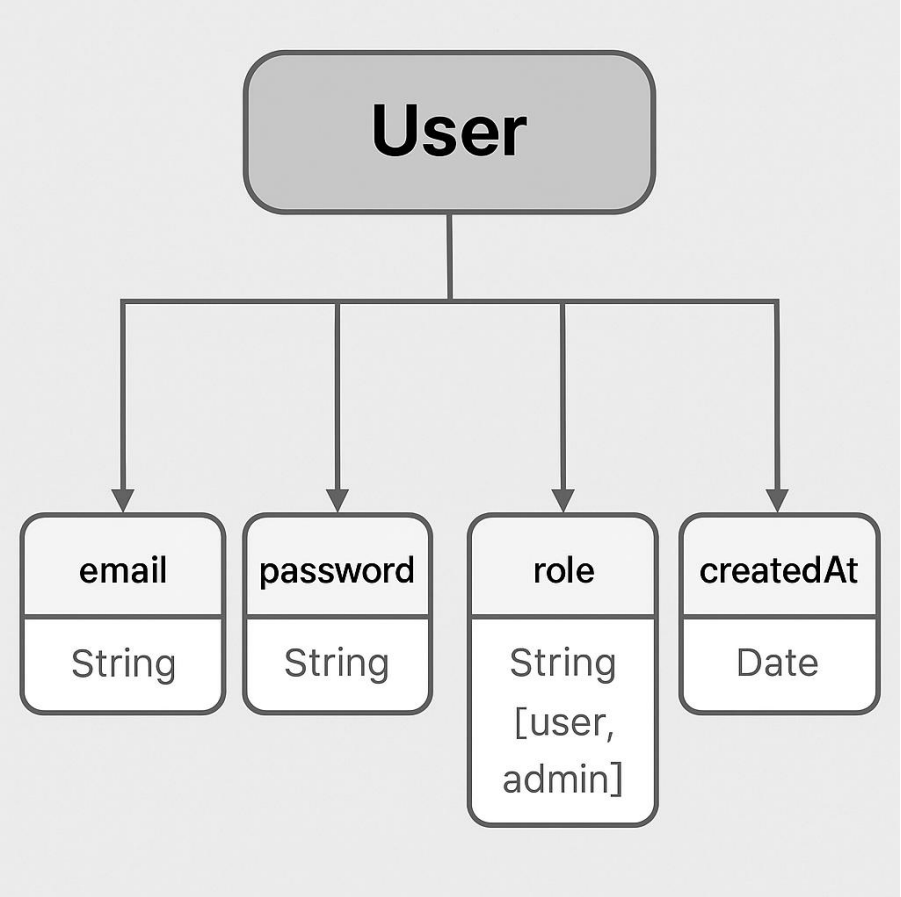

Рисунок 2.10 – Структура моделі User

Для підтримки клієнтської частини, яка працює на іншому порту (наприклад, React на 3000), реалізовано підтримку CORS (Cross-Origin Resource Sharing). Запити від клієнта обробляються у форматі JSON, а сервер відповідає відповідними даними або повідомленнями про помилки [19].

Додатково реалізовано просту валідацію введених даних та обробку стандартних помилок з поверненням відповідних кодів HTTP та повідомлень (наприклад, 401 – неавторизований доступ, 404 – не знайдено, 500 – помилка сервера).

Клієнтська частина надсилає запити через бібліотеку Axios, які обробляються Express-сервером. Усі запити мають зрозумілу структуру, відповідну REST-стандарту, та повертають JSON-відповіді, які можуть бути використані на фронтенді без додаткових трансформацій.

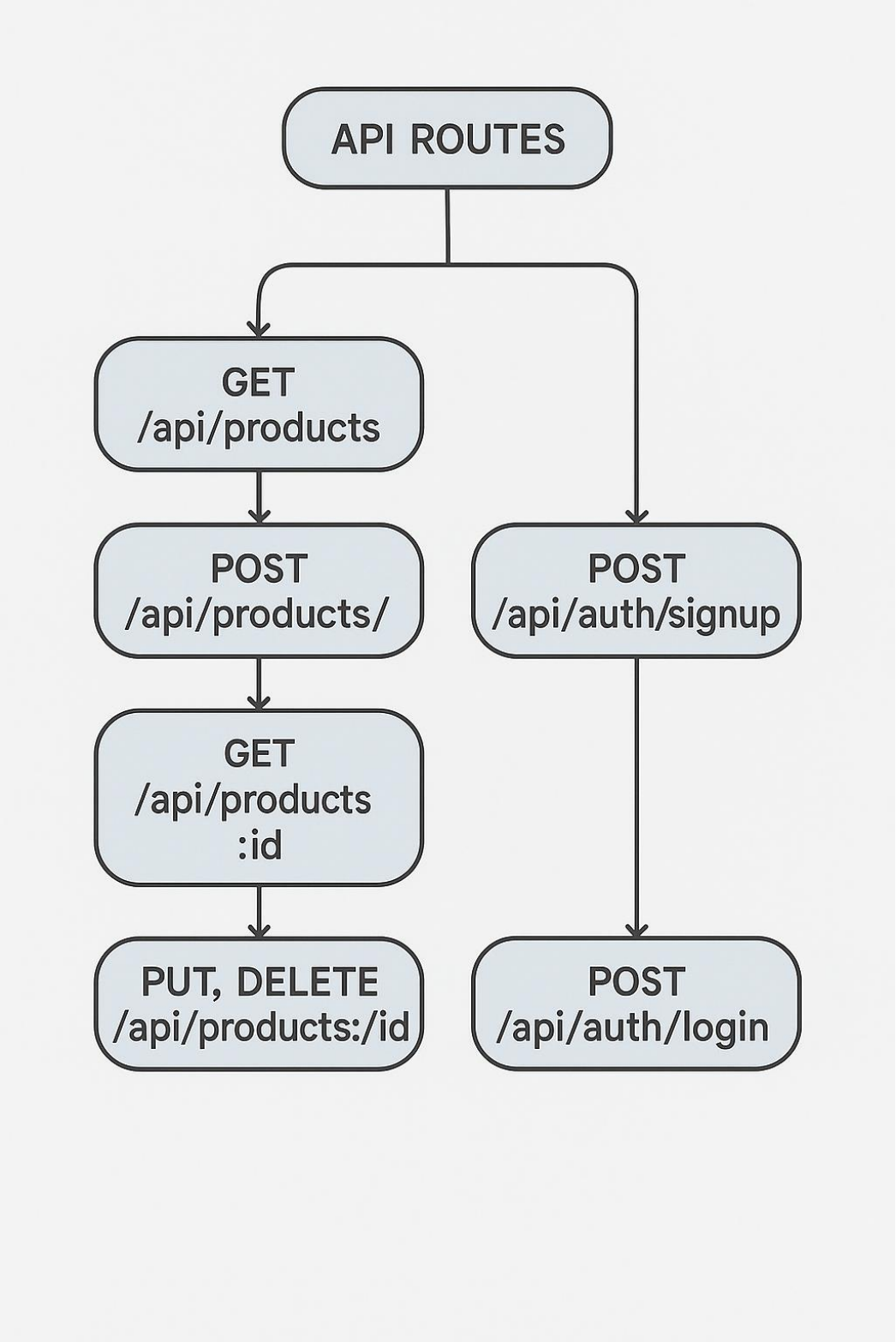

Рисунок 2.11 – Схема роботи маршрутів API

2.4 Висновок до розділу 2

У даному розділі розглянуто повний цикл розробки програмного модуля веб-ресурсу для інтернет-магазину електроніки – від проектування архітектури

системи до реалізації її клієнтської та серверної частин. Результати, досягнуті на кожному етапі, забезпечили побудову повноцінного веб-додатку, що відповідає сучасним вимогам до швидкості, функціональності, адаптивності та безпеки.

На етапі проектування було визначено логічну та фізичну структуру майбутньої системи, включаючи базові модулі, зв'язки між ними та ключові принципи взаємодії. Зокрема, було розроблено архітектуру для взаємодії між фронтендом та бекендом, а також описано структуру бази даних та моделі користувачів, продуктів та категорій. Архітектурний підхід передбачав використання клієнт-серверної моделі з чітким розподілом обов'язків між компонентами фронтенду та бекенду, що позитивно вплинуло на масштабованість та зручність подальшого обслуговування коду.

Клієнтська частина була реалізована за допомогою бібліотеки React, що дозволило побудувати компонентну структуру додатку з чітким розділенням логіки та представлення даних. Було створено набір функціональних компонентів, що включає форму додавання товару, список товарів з фільтрами, адміністративну панель з можливістю редагування та видалення елементів, а також модулі реєстрації та авторизації. Всі компоненти взаємодіють з API за допомогою бібліотеки Axios, що забезпечує зручну реалізацію асинхронних запитів.

Серверна частина розроблена з використанням Node.js та фреймворку Express, що забезпечило гнучкість та високу продуктивність під час обробки запитів. Серверна логіка забезпечує обробку CRUD-операцій над товарами, категоріями та користувачами, а також валідацію вхідних даних. Модель даних структурована відповідно до вимог NoSQL підходу, що дозволило легко адаптувати сховище до змін у структурі товарів та категорій.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-РЕСУРСУ ІНТЕРНЕТ-МАГАЗИНУ ЕЛЕКТРОНІКИ

3.1 Обґрунтування вибору стеку технологій

Під час розробки веб-ресурсу для інтернет-магазину електроніки особлива увага приділялася вибору оптимального технологічного стеку, який би забезпечував високу продуктивність, гнучкість, масштабованість, простоту підтримки та активну підтримку спільноти. Було проаналізовано низку альтернативних варіантів і на основі результатів сформовано наступний технологічний стек: React, Tailwind CSS, React Router, Node.js, Express.js, MongoDB, Mongoose, Axios та інші допоміжні інструменти.

React – це одна з найпопулярніших бібліотек JavaScript, розроблена Meta для створення сучасних користувацьких інтерфейсів. Головною перевагою React є його компонентна архітектура, яка дозволяє розбити інтерфейс на незалежні, повторно використовувані частини. Це значно полегшує розробку, тестування та масштабування проекту. Крім того, React використовує віртуальний DOM, який забезпечує ефективне оновлення інтерфейсу без необхідності повного перерендерингу сторінки, що, у свою чергу, покращує продуктивність програми.

Ще однією значною перевагою React є велика кількість готових бібліотек, документації та активна спільнота, що дозволяє швидко вирішувати технічні проблеми та легко знаходити рішення для реалізації нових функцій.

Для стилізації веб-ресурсу було обрано Tailwind CSS – утилітарний CSS-фреймворк, який дозволяє створювати інтерфейси за допомогою готових класів без написання власних CSS-стилів. Це значно пришвидшує процес верстки та дозволяє зберегти код чистим та структурованим. Tailwind CSS також підтримує адаптивний дизайн за замовчуванням, що критично важливо для сучасних сайтів, які повинні коректно відображатися на різних пристроях – від комп’ютерів до смартфонів [20].

Для реалізації маршрутизації в клієнтській частині використовується React Router, який дозволяє створювати SPA (Single Page Application), де перемикання

між сторінками не вимагає перезавантаження. Це покращує взаємодію користувача з сайтом та робить роботу з веб-ресурсом більш плавною.

Axios – бібліотека для HTTP-запитів, яка дозволяє зручно взаємодіяти з сервером. Axios підтримує обробку помилок, перехоплення запитів та відповідей, встановлення тайм-аутів та роботу з токенами, що важливо при реалізації функцій авторизації.

Для серверної частини було обрано Node.js, середовище виконання JavaScript, яке дозволяє виконувати код поза браузером. Node.js є надзвичайно популярним інструментом у веб-розробці, оскільки він дозволяє створювати високопродуктивні та масштабовані серверні рішення завдяки своїй асинхронній, неблокуючій архітектурі.

Express.js було обрано на основі Node.js, легкого та гнучкого фреймворку, який спрощує створення веб-додатків та RESTful API. Express дозволяє ефективно обробляти маршрути, запити та відповіді, а також інтегрувати проміжне програмне забезпечення, що зручно для реалізації контролю доступу, обробки даних JSON, автентифікації тощо.

Для зберігання даних використовується MongoDB, нереляційна база даних (NoSQL), яка дозволяє зберігати інформацію у вигляді документів формату BSON (JSON-like). MongoDB дозволяє зберігати неструктуровані дані, що робить його особливо зручним для реалізації таких об'єктів, як продукти з унікальними характеристиками або користувачі з гнучкою структурою [21].

Для інтеграції MongoDB з Node.js було використано бібліотеку Mongoose, яка реалізує підхід ORM (Об'єктно-реляційне відображення). Mongoose дозволяє описувати структуру даних за допомогою схем, встановлювати типи, значення за замовчуванням, правила перевірки та створювати індекси. Це значно підвищує контроль над даними та зменшує кількість потенційних помилок у логіці програми.

Особливу увагу при виборі стеку було приділено сумісності між компонентами. Усі вибрані інструменти добре інтегруються один з одним, мають активну спільноту, регулярно оновлюються та мають велику документацію. Стек

на основі JavaScript (React, Node.js та MongoDB) є типовим для повноцінного стеку MERN (MongoDB, Express, React, Node.js), який зарекомендував себе в тисячах комерційних проектів по всьому світу.

3.2 Порівняння альтернатив

У процесі вибору технологій для реалізації веб-ресурсу для інтернет-магазину електроніки було розглянуто кілька альтернативних рішень, кожне з яких має свої переваги та недоліки. Правильний вибір технологій суттєво впливає як на швидкість та ефективність розробки, так і на стабільність, масштабованість та зручність подальшої підтримки проекту.

У таблиці 3.1 наведено порівняльну характеристику основних фреймворків для розробки клієнтської частини.

Таблиця 3.1 – Порівняльна характеристика фреймворків клієнтської частини

Назва фреймворку	**Переваги**	**Недоліки**
Angular	Повна екосистема, типізація	Високий поріг входження
React	Гнучкість, велика спільнота	Потрібна додаткова конфігурація
Vue.js	Простота використання, низький поріг входження	Менш поширений у великих проектах

Для клієнтської сторони розглядалися такі фреймворки, як Angular, Vue.js та React. Angular – фреймворк з широкими можливостями, але його поріг входу досить високий, особливо для невеликих команд або окремих розробників. Vue.js вважається легшим у вивченні, але має менше сторонніх бібліотек та менш

розвинену екосистему. React забезпечує баланс між гнучкістю, простотою та широкими можливостями, що було вирішальним фактором на його користь.

Для вибору інструменту стилізації інтерфейсу було проаналізовано кілька популярних варіантів. Порівняльні особливості наведено в таблиці 3.2.

Таблиця 3.2 – Порівняльна характеристика систем стилізації

Назва системи	**Простота використання**	**Гнучкість налаштувань**	**Швидкість розробки**	**Необхідність додаткових знань**
CSS	Середня	Висока	Низька	Базова
Bootstrap	Висока	Середня	Висока	Мінімальна
Tailwind CSS	Висока	Висока	Висока	Середня

Альтернативами системи стилізації були Bootstrap та SCSS. Bootstrap – популярний фреймворк з готовими компонентами, але він накладає певні обмеження на унікальний дизайн. SCSS дозволяє гнучко створювати власні стилі, але вимагає більше часу на розробку. Tailwind CSS, який був обраний, дозволяє швидко та зручно створювати адаптивні інтерфейси завдяки своєму утилітарному підходу, не жертвуючи налаштуванням.

Порівняно з класичним CSS, Tailwind CSS значно пришвидшує процес розробки завдяки готовим допоміжним класам, які дозволяють створювати інтерфейс без написання великої кількості власного коду. У той час як використання звичайного CSS вимагає більшої кількості окремих селекторів, ручного створення класів та структурування стилів у зовнішніх файлах, Tailwind дозволяє реалізовувати стиль безпосередньо в HTML-розмітці.

Як показано в таблиці 3.3, фреймворк Node.js виявився найоптимальнішим для реалізації серверної частини, оскільки він забезпечує високу продуктивність, легку інтеграцію з MongoDB та активну підтримку з боку спільноти.

Таблиця 3.3 – Порівняльна характеристика засобів розробки серверної частини

Назва фреймворку	**Мова програмування**	**Поширення**	**Підтримка MongoDB**	**Продуктивність**
Node.js	JavaScript	Дуже популярний	Добра (через Mongoose)	Висока
Django	Python	Популярний	Не основна, потрібні обхідні рішення	Середня
Laravel	PHP	Популярний	Можлива, але не типова	Середня

Серед розглянутих серверних технологій були Node.js, PHP та Django. PHP традиційно використовується для розробки веб-сайтів, але він поступається сучасним JavaScript-стекам у зручності створення REST API. Django, побудований на Python, є безпечним та ефективним рішенням, але його інтеграція з фронтендом React вимагає додаткових налаштувань. Обраний варіант – Node.js з Express.js, який дозволяє зручно реалізувати серверну логіку, залишаючись у межах одного мовного середовища (JavaScript), що спрощує як розробку, так і підтримку проектів та обраної бази даних MongoDB.

Як видно з таблиці 3.4, MongoDB є найбільш доцільним вибором для даного проекту, оскільки дозволяє зберігати структуровану інформацію у вигляді документів, що відповідає структурі об'єктів у JavaScript.

Таблиця 3.4 – Порівняльна характеристика систем керування базами даних

Назва БД	**Тип БД**	**Структура даних**	**Складність запитів**	**Взаємодія з Node.js**
MongoDB	Документоорієнтована (NoSQL)	BSON-документи	Відносно проста	Відмінна (через Mongoose)
MySQL	Реляційна (SQL)	Таблиці	Висока	Через ORM / драйвери
PostgreSQL	Реляційна (SQL)	Таблиці	Висока	Через ORM / драйвери

Альтернативами були MySQL та PostgreSQL, які є реляційними СУБД. Їхніми сильними сторонами є підтримка складних запитів та транзакцій, але вони вимагають чітко визначеної структури таблиць та суворого дотриманя схем. Для проекту, де структура даних є гнучкою (наприклад, продукти з різними властивостями), більш підходящою виявилася MongoDB, документоорієнтована СУБД, яка дозволяє зберігати дані у форматі BSON (JSON-подібному). Вона краще підходить для динамічних застосунків, де структура може змінюватися.

В результаті аналізу представлених альтернатив було зроблено висновок, що обраний технологічний стек є оптимальним для реалізації поставлених завдань. Він забезпечує високу гнучкість під час розробки, спрощує взаємодію між клієнтською та серверною частинами, а також відповідає сучасним вимогам до продуктивності, масштабованості та підтримки. Вибір React для фронтенду дозволяє легко розширювати функціональність та підтримувати модульну архітектуру, а використання Tailwind CSS значно скорочує час на розробку інтерфейсу, залишаючи достатньо місця для індивідуального дизайну.

На стороні сервера платформа Node.js у поєднанні з Express.js демонструє хорошу продуктивність під час обробки запитів та дозволяє підтримувати єдиний технологічний стек JavaScript. MongoDB як СУБД стала хорошим вибором для

зберігання структурованих, але різнорідних даних, які часто зустрічаються в електронній комерції – наприклад, товари з різними характеристиками залежно від категорії. Така гнучкість зменшує потребу в складному дизайні таблиць та дозволяє швидше адаптувати систему до змін у бізнес-логіці.

Таким чином, врахування технічних, функціональних та організаційних аспектів під час вибору інструментів дозволило сформувати збалансований підхід до побудови веб-ресурсу, який поєднує в собі простоту розробки, ефективність у роботі та перспективи подальшого розвитку.

3.3 Створення WEB-ресурсу

Розробка програмного модуля для інтернет-магазину електроніки відбувалася поетапно, дотримуючись принципів модульності, легкості розширення та масштабованості. Основу проекту було побудовано на стеку MERN (MongoDB, Express.js, React, Node.js).

Інтерфейс користувача реалізовано з використанням React, що забезпечує високу продуктивність і динамічність. Для стилізації застосовано Tailwind CSS, що дозволило гнучко керувати адаптивністю компонентів. Основною точкою входу є компонент App.jsx, який відповідає за маршрутизацію за допомогою бібліотеки react-router-dom.

```
<Route path="/cart" element={<Cart />} />
<Route path="/orders" element={<Orders />} />
<Route path="/admin/orders" element={<AdminOrders />} />
```

Як можна спостерігати з цього блоку коду, для кожного шляху визначено окремий компонент, наприклад, /cart, /login, /orders тощо.

Застосовано контекст AuthContext, який зберігає інформацію про користувача та забезпечує авторизацію на основі токена jsx, як зображено на фрагменті коду:

```
const login = async (email, password) => {
      const res = await axios.post('/api/auth/login', { email,
      password });
      localStorage.setItem('token', res.data.token);
      setUser(res.data.user);
};
```

Також було передбачено умовну навігацію: залежно від ролі користувача відображаються кнопки "Кошик" та "Мої замовлення" або для адміністратора, "Користувачі" та "Замовлення".

```
const res = await
axios.post('http://localhost:5000/api/auth/register', formData);
```

У цьому фрагменті POST-запит надсилається на сервер для реєстрації нового користувача. Для зручної роботи з HTTP-запитами ми використовуємо бібліотеку axios. Дані форми, що містять електронну пошту, пароль та роль (наприклад, user), передаються в тілі запиту на маршрут /api/auth/register [22].

На стороні сервера запит обробляється контролером, який перевіряє унікальність електронної пошти, хешує пароль за допомогою бібліотеки bcrypt та створює нового користувача в базі даних MongoDB. Після успішної реєстрації клієнту повертається повідомлення про успіх або відповідна помилка у разі неправильного введення.

```
const res = await axios.get('/api/products');
setProducts(res.data);
```

Цей код відповідає за отримання даних про всі доступні товари в магазині. Метод axios.get() виконує GET-запит до маршруту /api/products, де сервер повертає JSON-масив з інформацією про всі товари, що містяться в базі даних.

Після отримання відповіді функція setProducts() оновлює стан компонента React, передаючи нові дані до списку товарів, що відображаються на головній сторінці. Це забезпечує динамічне оновлення контенту без перезавантаження сторінки.

```
const res = await axios.post('/api/auth/login', formData);
localStorage.setItem('token', res.data.token);
```

Згідно з даним фрагментом коду, після натискання кнопки "Увійти", дані форми надсилаються на сервер для перевірки. У разі правильного логіну й паролю сервер генерує JWT-токен, який повертається у відповіді.

Отриманий токен зберігається у localStorage браузера, що дозволяє зберігати стан авторизації між оновленнями сторінки. Цей токен буде надсилатися у заголовку Authorization при кожному запиті, де потрібна автентифікація, наприклад, для перегляду замовлень.

```
const res = await axios.post('/api/products', newProduct, {
      headers: { Authorization: `Bearer ${token}` }
});
```

Показана функція дозволяє адміністратору додавати нові товари через форму. У тілі запиту передається об’єкт newProduct, що містить назву, опис, ціну, посилання на зображення та категорію. У заголовку додається токен адміністратора, що підтверджує його права доступу.

На сервері цей запит потрапляє під middleware для перевірки автентифікації та авторизації, після чого відповідний контролер створює новий запис у базі даних. У разі успішного створення товару повертається об’єкт із новим товаром.

```
await axios.post('/api/orders', { items, total }, {
```

```
  headers: { Authorization: `Bearer ${token}` }
});
```

Наданий фрагмент коду відповідає за опцію "Оформити замовлення", при натисканні на дану кнопку відбувається запит на створення нового замовлення. Об’єкт items містить список товарів, а total – загальну суму замовлення. Токен, що передається в заголовку, дозволяє серверу ідентифікувати користувача.

На сервері створюється новий документ замовлення, до якого додається userId, отриманий із токена. Це дозволяє у майбутньому отримувати список замовлень конкретного користувача або виконувати дії від імені адміністратора.

```
const res = await axios.get('/api/orders/admin/orders', {
  headers: { Authorization: `Bearer ${token}` }
});
```

На наданому блоку коду показано запит замовлень з сервера. Адміністратор може переглядати всі замовлення на сайті. Цей запит надсилається з токеном адміністратора, а сервер повертає список замовлень із доданими даними користувача (через populate в Mongoose).

Цей механізм дозволяє зручно адмініструвати магазин, перевіряти історію покупок, редагувати або видаляти замовлення та надавати технічну підтримку за потреби.

У даному підрозділі було описано процес створення веб-ресурсу. Використання бібліотеки axios для HTTP-запитів клієнтів та використання localStorage для зберігання токенів автентифікації забезпечує зручну взаємодію між фронтендом та бекендом. Особливу увагу було приділено захисту маршрутів за допомогою JWT, що дозволяє розділити права доступу між звичайними користувачами та адміністраторами.

Реалізовано основні функції, необхідні для роботи магазину: відображення товарів, додавання нових товарів адміністратором, розміщення замовлень

користувачем та перегляд історії покупок. З боку адміністратора також реалізовано можливість перегляду, редагування та видалення замовлень, а також їх фільтрації за ключовими параметрами. Всі запити на захищені маршрути обробляються з урахуванням ролі користувача.

Таким чином, етап створення основного функціоналу системи завершено. Після реалізації всіх запланованих модулів та логіки взаємодії між клієнтською та серверною частинами, ми переходимо до наступного етапу – тестування веб-ресурсу для перевірки коректності роботи програмних модулів, виявлення можливих помилок та оцінки зручності використання користувацького інтерфейсу.

3.4 Тестування WEB-ресурсу

Після завершення реалізації клієнтської та серверної частин було проведено функціональне тестування створеного веб-ресурсу інтернет-магазину електроніки. Основною метою тестування є перевірка працездатності основних функцій системи та виявлення можливих помилок, які могли виникнути під час розробки.

Першим етапом тестування була перевірка відображення головної сторінки для неавторизованого користувача. Було підтверджено, що користувач має доступ лише до функцій входу та реєстрації. Для отримання доступу до основних функцій веб-ресурсу користувачу необхідно пройти реєстрацію або авторизуватись.

Наступним етапом була перевірка правильності маршрутизації та доступності головних сторінок, зокрема каталогу товарів, сторінки входу, реєстрації та кошика. Переходи між сторінками здійснюються без затримок та перезавантажень, що відповідає очікуваній поведінці SPA-додатку. Тестування буде продовжено з акцентом на перевірку ролей користувачів, динамічну обробку даних та роботу з бекендом через API.

Як показано на рисунку 3.1, інтерфейс є зрозумілим, користувач бачить логотип магазину, форму авторизації та опцію реєстрації.

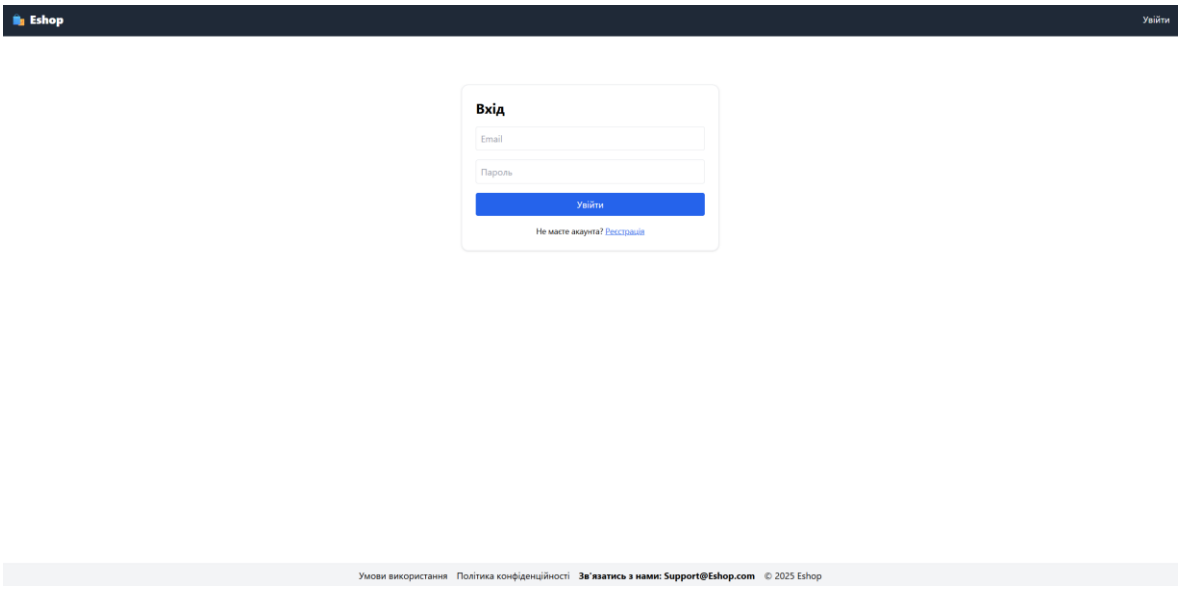

Рисунок 3.1 – Головна сторінка веб-ресурсу для неавторизованого користувача

Після реєстрації та авторизації відкривається доступ до функцій перегляданння каталогу товарів, додавання товарів до кошика, перегляду кошика та оформлення замовлення.

Як показано на рисунку 3.2, у кошику відображаються товари, які користувач додав із каталогу, вказується їхня назва, вартість кожного товару та загальна вартість. Також показано функції очищення кошика, видалення окремого товару та оформлення замовлення. При видаленні окремого товару сума також змінюється залежно від товарів, які залишились в кошику. Адміністратор також може змінювати загальну суму замовлення без видалення товару з замовлення.

Крім того, реалізовано контроль доступу – адміністратор має окрему панель керування, недоступну для звичайних користувачів. Усі дії користувача супроводжуються відповідними повідомленнями про успіх або помилку, що покращує загальну взаємодію з системою. Інтерфейс адаптований до мобільних пристроїв, що забезпечує комфортне використання веб-ресурсу незалежно від розміру екрана.

Також було протестовано функціональність оформлення замовлення. Після натискання кнопки «Оформити замовлення» дані успішно надсилаються

на сервер, що підтверджується повідомленням про успішну операцію. Кошик очищається одразу після оформлення замовлення.

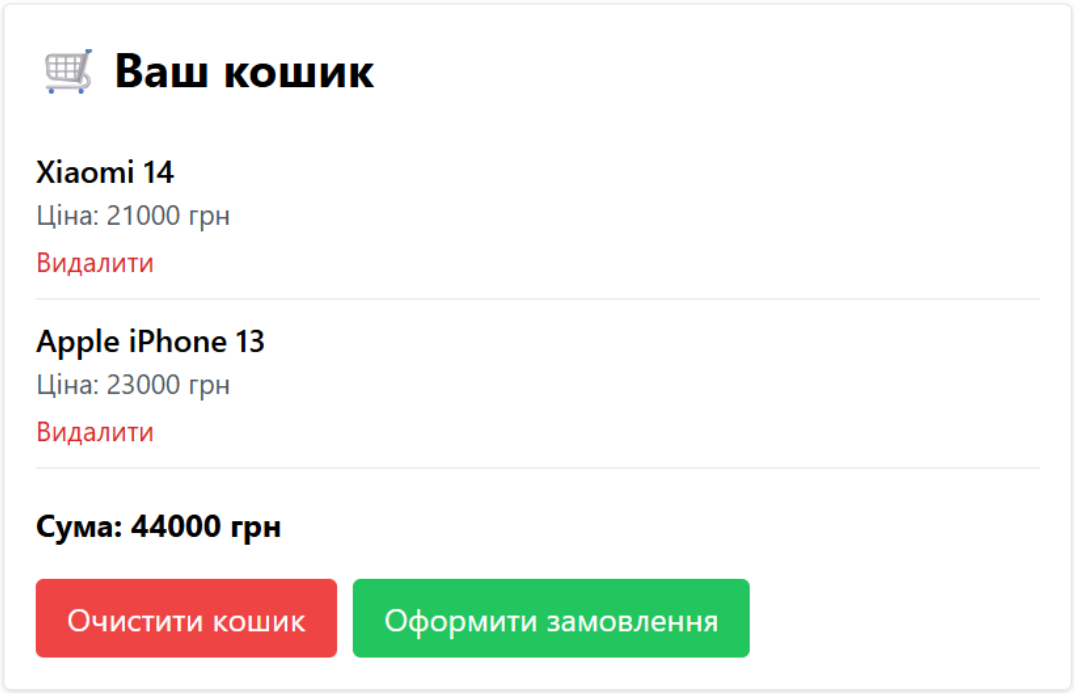

Рисунок 3.2 – Кошик користувача з доданими товарами

На рисунку 3.3 зображено повідомлення, яке бачить користувач після завершення оформлення.

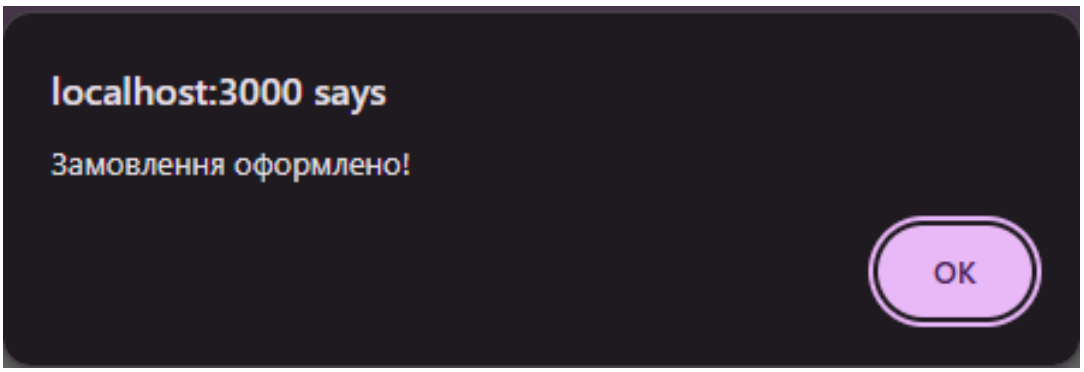

Рисунок 3.3 – Повідомлення про успішне оформлення замовлення

Після здійснення замовлення користувач зможе переглянути свої замовлення, натиснувши на опцію «Мої замовлення» зверху на сторінці веб-

сайту. На сторінці з замовленнями користувача буде показано усі здійсненні замовлення з вказанням номера (ID) замовлення, датою та часом замовлення, загальною сумою замовлення, а також списком товару з вказанною назвою товару та ціною кожного товару окремо.

На рисунку 3.4 показано меню замовлень користувача, на якому видно, що вся необхідна інформація відображається коректно.

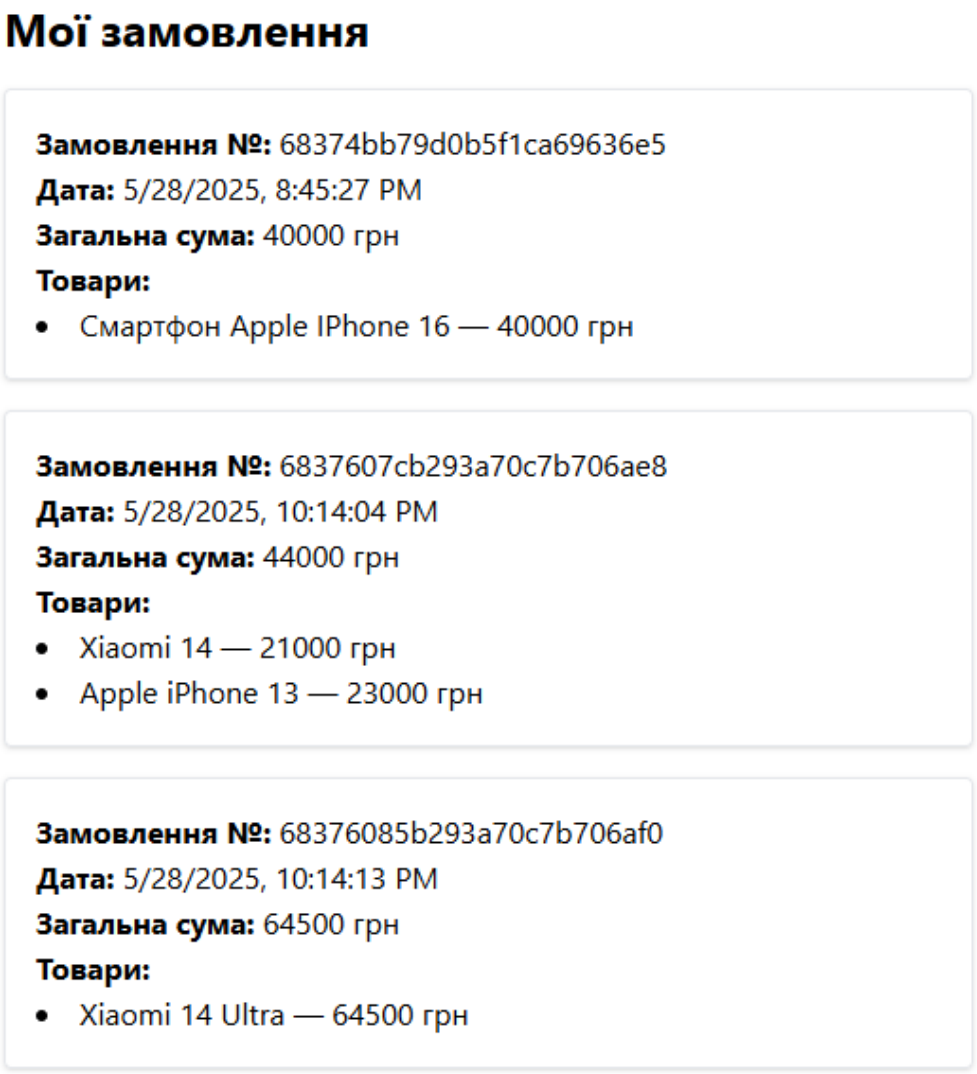

Рисунок 3.4 – Меню замовлень користувача

Для адміністратора доступна окрема панель керування, яка дозволяє додавати нові товари, редагувати або видаляти існуючі, а також переглядати список усіх замовлень і користувачів.

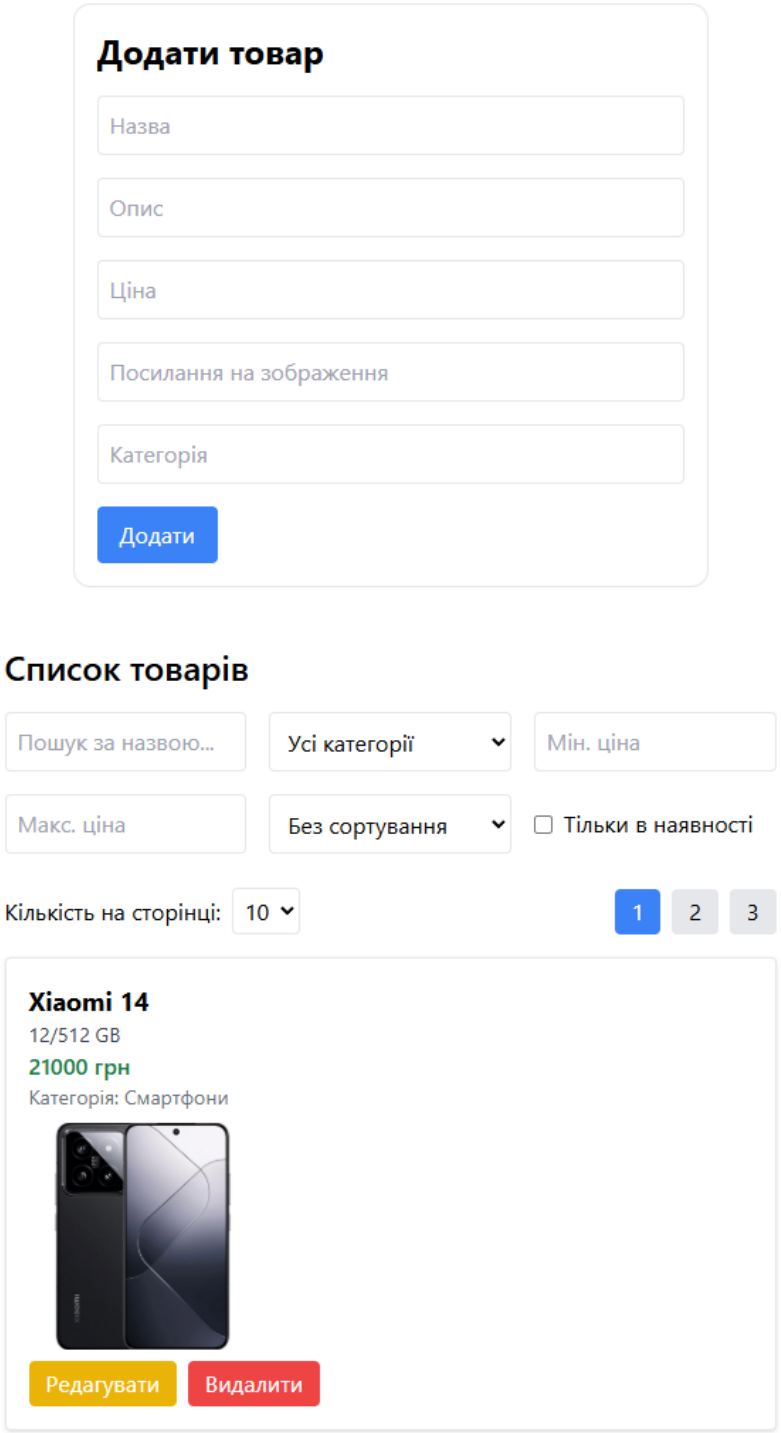

Рисунок 3.5 – Адміністративна панель управління товарами

Як видно з рисунку 3.5, адміністратор може керувати товарами через просту форму з відповідними полями.

Також адміністратор може переглядати повний список замовлень за вказаними фільтрами (електронна пошта користувача, назва товару, дата замовлення), редагувати повну суму замовлення та видаляти замовлення або окремий товар за потреби як показано на рисунку 3.6.

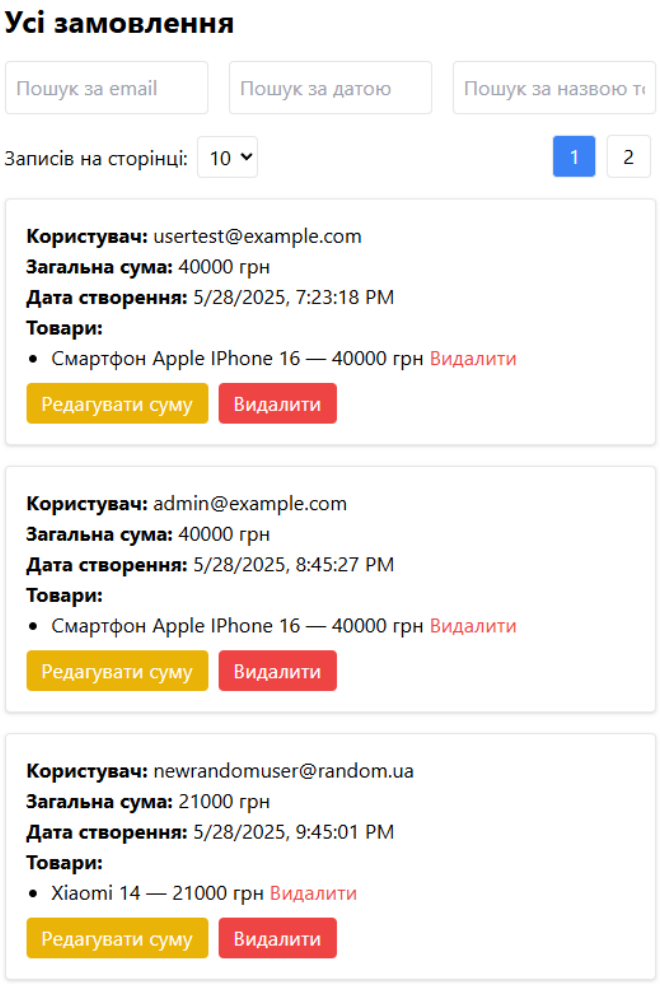

Рисунок 3.6 – Перегляд усіх замовлень в адміністративній панелі

3.5 Особливості впровадження та перспективи розвитку

Розроблений веб-ресурс інтернет-магазину електроніки має всі необхідні компоненти для реалізації в реальному середовищі. Клієнтська частина, реалізована за допомогою бібліотеки React та стилізована за допомогою Tailwind

CSS, забезпечує адаптивний дизайн, який коректно відображається на різних пристроях та браузерах. Серверна частина, створена на платформі Node.js з використанням фреймворку Express, є ефективною з точки зору обробки запитів та масштабування. База даних MongoDB дозволяє зручно зберігати структуровану інформацію про товари, користувачів та замовлення.

Для повноцінної реалізації необхідно виконати низку технічних дій. Клієнтська частина повинна бути розгорнута на статичному хостингу, такому як Netlify або Vercel. Серверна частина може бути розгорнута на Heroku, Render або VPS, залежно від навантаження та бюджету. Для бази даних доцільно використовувати MongoDB Atlas або локальний сервер MongoDB з відповідною конфігурацією. Також необхідно налаштувати змінні середовища через файл .env, реалізувати систему автоматичного резервного копіювання та ввімкнути моніторинг продуктивності сервера [23].

Реалізований інтернет-магазин має потенціал для комерційного використання, наприклад, як MVP-версія стартапу. Наявність адміністративної панелі з функціями управління товарами дозволяє ефективно працювати з асортиментом та оновлювати його в режимі реального часу. Це важливий крок до повноцінної системи управління контентом всередині магазину [24].

У майбутньому проект можна розширити за рахунок інтеграції з платіжними системами. Наприклад, підключення LiqPay, Fondy або Stripe дозволить впровадити онлайн-оплату товарів та відстежувати замовлення. Це значно розширить функціональність ресурсу та дозволить обслуговувати клієнтів дистанційно без необхідності фізичного контакту.

Ще одним доцільним кроком було б впровадження реєстрації користувачів через сторонні сервіси за протоколом OAuth. Можливість входу через облікові записи Google, Facebook або GitHub спростить авторизацію для користувачів і водночас підвищить безпеку облікових даних.

Окрему увагу можна приділити впровадженню особистого облікового запису користувача. Буде зручно переглядати історію замовлень, редагувати

профіль, залишати відгуки та керувати обраними товарами. Це покращить взаємодію з клієнтом та зробить користування сайтом зручнішим.

Для покращення навігації по каталогу товарів можна впровадити складну систему фільтрів та сортування. Наприклад, сортування за ціною, рейтингом чи популярністю, фільтрація за виробником або параметрами, такими як розмір пам'яті чи діагональ екрана. Це допоможе користувачам швидко знайти потрібний товар серед великої кількості пропозицій.

Ще одним напрямком розвитку є підключення аналітичних сервісів, таких як Google Analytics або Hotjar. Вони дозволяють відстежувати поведінку користувачів на сайті, що дає можливість покращити інтерфейс, виявити слабкі місця та скоригувати маркетингову стратегію.

З огляду на широке використання мобільних пристроїв, перспективним є створення мобільного застосунку на базі React Native. Це дозволить охопити ще більшу аудиторію, зокрема тих користувачів, які віддають перевагу мобільному шопінгу.

Також необхідно передбачити можливість багатомовної підтримки інтерфейсу. Реалізація сайту українською, англійською та іншими мовами сприятиме виходу на міжнародні ринки та підвищить доступність ресурсу для користувачів з різних країн [25].

Одним із важливих напрямків розвитку є впровадження двофакторної автентифікації (2FA). Це підвищить безпеку облікових записів користувачів, особливо для адміністративного доступу. Використання тимчасових кодів або авторизація через мобільні додатки допоможе запобігти несанкціонованому доступу.

Для підвищення привабливості ресурсу для користувачів доцільно буде розмістити рекламний банер на головній сторінці або інших ключових областях сайту. Це може бути як внутрішня реклама акцій магазину, так і інтеграція з партнерами для монетизації.

Інтеграція із соціальними мережами (Facebook, Instagram, Telegram) дозволить розширити маркетингові можливості. Зокрема, впровадження кнопок

поширення, відгуків, входу через соціальні акаунти, а також просування товарів через таргетовану рекламу забезпечить приплив нових користувачів.

Також важливо забезпечити постійну технічну підтримку користувачів. Вона може бути реалізована як у формі чату з оператором, так і через систему заявок або базу знань. Своєчасне реагування на запити клієнтів є запорукою підтримки лояльності та покращення репутації.

У майбутньому варто розглянути співпрацю з поштовими операторами (наприклад, Нова Пошта, Укрпошта), що дозволить не лише автоматизувати процес доставки, але й запропонувати клієнтам ширший спектр логістичних варіантів, включаючи кур'єрську доставку до дверей та самовивіз з поштових відділень.

Таким чином, веб-ресурс не лише виконує свою основну функцію, але й має широкий спектр можливостей масштабування, адаптації до реальних потреб ринку та подальшого розвитку. У майбутньому цей проект може стати основою для повноцінної комерційної платформи з підтримкою електронної комерції, логістики та маркетингу.

3.6 Висновок до розділу 3

В даному розділі було здійснено повний цикл розробки веб-ресурсу інтернет-магазину електроніки: від вибору технологічного стеку до реалізації ключових функціональних модулів.

У підрозділі 3.1 обґрунтовано вибір технологій - React для фронтенду, Node.js та Express для бекенду та MongoDB для бази даних. Такий стек забезпечує високу продуктивність, масштабованість та гнучкість у реалізації функціоналу.

У підрозділі 3.2 проведено порівняння альтернативних технологій та доведено переваги обраного підходу з урахуванням поставлених цілей у рамках розробки MVP функціонального інтернет-магазину.

Підрозділ 3.3 присвячено реалізації функціональних можливостей: створено адаптивну клієнтську частину, розроблено REST API для обробки

запитів, реалізовано авторизацію з розділенням прав, додано функціонал для розміщення замовлень, перегляду кошика, а також адміністративну панель для управління користувачами та замовленнями. Фрагменти коду з відповідними описами демонструють ключові аспекти реалізації.

У підрозділі 3.4 було проведено тестування веб-ресурсу. За допомогою скріншотів та описів було перевірено коректність роботи головних сторінок, форм, системи замовлень та функціональності для адміністратора. Усі ключові компоненти пройшли функціональне тестування без критичних помилок.

У підрозділі 3.5 окреслено особливості впровадження веб-ресурсу в реальному середовищі: визначено етапи розгортання клієнтської та серверної частин, а також описано перспективи розвитку проекту, включаючи інтеграцію платіжних систем, багатомовність, мобільний додаток, аналітику, маркетинг та підтримку користувачів.

Впроваджений веб-ресурс повністю відповідає вимогам та може слугувати основою для запуску реального інтернет-магазину з подальшим масштабуванням. Після завершення розробки та початкового тестування наступним етапом є проведення розширеного функціонального тестування, оптимізація продуктивності та підготовка до впровадження у виробничому середовищі.

ВИСНОВКИ

У бакалаврській роботі досліджено та розроблено веб-ресурс для інтернет-магазину електроніки. Під час аналізу предметної області було визначено доцільність розробки сучасного функціонального веб-додатку, який би забезпечував зручну взаємодію користувача з онлайн-платформою для продажу техніки. Було проаналізовано існуючі програмні рішення та визначено ключові компоненти, які слід впровадити в систему для забезпечення її конкурентоспроможності.

Усі задачі, поставлені перед бакалаврською кваліфікаційною роботою виконані в повному обсязі, а саме:

- аналіз предметної області електронної комерції та відомих програм аналогів;
- постановка задачі для створення веб-ресурсу інтернет-магазину електроніки;
- проектування системи веб-ресурсу інтернет-магазину електроніки;
- розробка клієнтської та серверної частини веб-ресурсу інтернет-магазину електроніки;
- обґрунтування вибору стеку технологій та порівняння альтернатив веб-ресурсу інтернет-магазину електроніки;
- створення та тестування модулів веб-ресурсу інтернет-магазину електроніки.

Повноцінна клієнтська частина була реалізована за допомогою React та Tailwind CSS, а також серверна логіка на Node.js з Express та базою даних MongoDB. Адміністратор може керувати товарами, переглядати та редагувати замовлення, фільтрувати та шукати за різними параметрами. Сторінки адаптовані для мобільних пристроїв та забезпечують зручну навігацію.

Розроблений веб-ресурс підтвердив свою технічну ефективність у рамках освітнього проекту, має всі передумови для масштабування, реального

впровадження та комерційного використання в майбутньому. Розроблений інтернет-магазин може бути використаний як стартова комерційна платформа або MVP для стартапу в сфері онлайн-торгівлі, що підвищує продуктивність вебресурсу інтернет магазину електроніки. У майбутньому проект може бути розширений шляхом інтеграції платіжних сервісів, мобільного додатку, підтримки кількох мов, соціальних функцій та підключення логістичних служб.

Мета розробки – розширення функціональних можливостей веб-ресурсу інтернет магазину електроніки за рахунок покращення продуктивності ресурсу досягнута.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Є.С. Кривовязюк, В.О. Денисюк. Розробка WEB-ресурсу інтернет-магазину електроніки. Молодь в науці: дослідження, проблеми, перспективи (МН-2025). Вінницький національний технічний університет. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/24568/20344
2. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. Вінниця: ВНТУ, 2023. (PDF, 54 с.). URL: https://iq.vntu.edu.ua/method/getfile.php?fname=124285.pdf&x=1&card_id=46522&id=124285
3. Бурба В.П., Гриценко В.Г. Основи веб-програмування. – К.: КНУ, 2020. – 285 с.
4. Литвин О.М. Основи баз даних: навчальний посібник. – Львів: ЛНУ, 2019. – 215 с.
5. Власюк В. Web-дизайн і розробка сайтів. – Тернопіль: ТНТУ, 2018. – 156 с.
6. Власюк І. Сучасні підходи до UI/UX у веброзробці // Технології в інформаційних системах. – 2022. – №1. – С. 17–22.
7. Глушак А.В. Технології проектування веб-систем: навч. посіб. – Вінниця: ВНТУ, 2021. – 147 с.
8. Шевчук С.П. Основи проектування інформаційних систем. – К.: Видавничий дім «Слово», 2021. – 289 с.
9. React documentation [Електронний ресурс]. URL: https://reactjs.org/docs/getting-started.html
10. Express.js documentation [Електронний ресурс]. URL: https://expressjs.com/
11. JWT.io. JSON Web Token Introduction [Електронний ресурс]. URL: https://jwt.io/introduction

12. Axios. Official documentation [Електронний ресурс]. URL: https://axios-http.com/docs/intro
13. Tailwind CSS documentation [Електронний ресурс]. URL: https://tailwindcss.com/docs
14. Фленаган Д. JavaScript. Повне керівництво. – Львів: Вільямс, 2021. – 976 с.
15. Шевчук А.В. Створення веб-додатків з використанням React // Вісник комп'ютерних наук. – 2021. – №4. – С. 33–41.
16. Нгуен В. Node.js для початківців. – К.: Діалектика, 2020. – 352 с.
17. Mongoose Documentation [Електронний ресурс]. URL: https://mongoosejs.com/docs/
18. Крамаренко С. Інформаційна безпека веб-додатків. – Харків: ХНУРЕ, 2020. – 203 с.
19. JSON Server [Електронний ресурс]. URL: https://github.com/typicode/json-server
20. CSS-Tricks. Flexbox Guide [Електронний ресурс]. URL: https://css-tricks.com/snippets/css/a-guide-to-flexbox/
21. MongoDB Manual [Електронний ресурс]. URL: https://www.mongodb.com/docs/manual/
22. Postman Learning Center [Електронний ресурс]. URL: https://learning.postman.com/
23. Stack Overflow Developer Survey 2023 [Електронний ресурс]. URL: https://survey.stackoverflow.co/2023
24. Маркетингові рішення в інтернет-магазинах // Економіка і бізнес. – 2021. – №2. – С. 91–97.
25. Web Accessibility Guidelines (WAI) [Електронний ресурс]. URL: https://www.w3.org/WAI/standards-guidelines/wcag/

ДОДАТОК А (ОБОВ'ЯЗКОВИЙ)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка WEB-ресурсу інтернет-магазину електроніки

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,96 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____ _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Денисюк В.О.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Кривовязюк Є.С.
(підпис) (прізвище, ініціали)

ДОДАТОК Б (ОБОВ'ЯЗКОВИЙ)

Лістинг програми

```
//../client/src/App.jsx
import { BrowserRouter as Router, Routes, Route, Navigate, useNavigate } from
'react-router-dom';
import { useAuth } from './context/AuthContext';
import { CartProvider, useCart } from './context/CartContext';
import Login from './components/Login';
import Register from './components/Register';
import AddProductForm from './components/AddProductForm';
import ProductList from './components/ProductList';
import ProductDetails from './components/ProductDetails';
import Cart from './components/Cart';
import Orders from './pages/Orders';
import Header from './components/Header';
import Footer from './components/Footer';
import Terms from './pages/Terms';
import Privacy from './pages/Privacy';
import UserList from './pages/UserList';
import AdminUsers from './components/AdminUsers';
import AdminOrders from './components/AdminOrders';

function AppWrapper() {
 return (
  <Router>
   <CartProvider>
    <App />
   </CartProvider>
  </Router>
```

```
  );
}

function App() {
  const { user, logout } = useAuth();
  const isAdmin = user?.role === 'admin';
  const navigate = useNavigate();
  const { cartItems } = useCart();
  const totalQuantity = cartItems.reduce((sum, item) => sum + (item.quantity || 1), 0);
  const handleLogout = () => {
    logout();
    navigate('/login');
  };

  return (
    <div>
      <Header />
      <div className="pt-[64px] pb-[60px] p-4 max-w-xl mx-auto">
        <Routes>
          <Route path="/login" element={<Login />} />
          <Route path="/register" element={<Register />} />
          <Route
            path="/"
            element={
              user ? (
                <>
                  {isAdmin && <AddProductForm />}
                  <ProductList />
                </>
              ) : (
```

```
            <Login />
          )
        }
      />
      {isAdmin && (
        <>
      <Route path="/admin/users" element={<AdminUsers />} />
      <Route path="/admin/orders" element={<AdminOrders />} />
        </>
      )}
      <Route path="/product/:id" element={<ProductDetails />} />
      <Route path="/cart" element={<Cart />} />
      <Route path="/orders" element={<Orders />} />
      <Route path="/terms" element={<Terms />} />
      <Route path="/privacy" element={<Privacy />} />
      <Route path="/users" element={isAdmin ? <UserList /> : <Navigate to="/" />}
      />
    </Routes>
  </div>
  <Footer />
 </div>
 );
}
export default AppWrapper;

//../server/server.js
import express from 'express';
import mongoose from 'mongoose';
import cors from 'cors';
import dotenv from 'dotenv';
```

```
import productRoutes from './routes/productRoutes.js';
import authRoutes from './routes/authRoutes.js';
import orderRoutes from './routes/orderRoutes.js';
import userRoutes from './routes/userRoutes.js';

dotenv.config();
const app = express();
const PORT = 5000;

// Middleware
app.use(cors());
app.use(express.json());

// Routes
app.use('/api/products', productRoutes);
app.use('/api/auth', authRoutes);
app.use('/api/orders', orderRoutes);
app.use('/api/users', userRoutes);

// MongoDB
mongoose.connect('mongodb://127.0.0.1:27017/eshop', {
  useNewUrlParser: true,
  useUnifiedTopology: true
})

.then(() => {
  console.log('MongoDB connected (local)');
  app.listen(PORT, () => {
    console.log(`Server running on http://localhost:${PORT}`);
  });
```

```
})
.catch(err => {
  console.error('MongoDB connection error:', err.message);
});
app.get('/', (req, res) => {
  res.send('API is running...');
});
```

ДОДАТОК В (ОБОВ'ЯЗКОВИЙ)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА WEB-РЕСУРСУ ІНТЕРНЕТ-МАГАЗИНУ ЕЛЕКТРОНІКИ

Виконав: студент 4 курсу,групи 3КН-21б
спеціальності 122 Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Кривовязюк Є.С.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Денисюк В.О.
(прізвище та ініціали)

«03» червня 2025 р.

Вінниця ВНТУ - 2025 рік

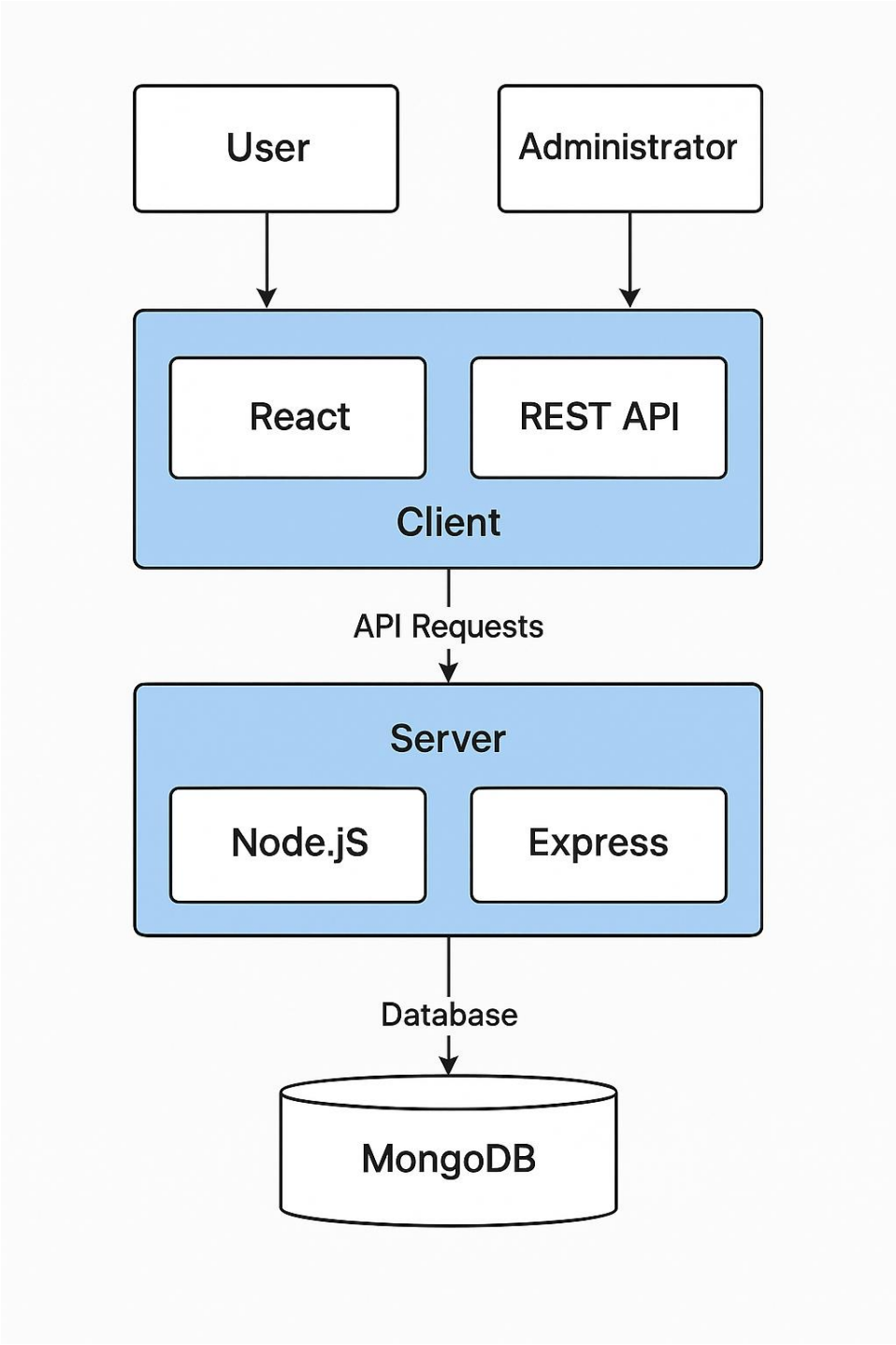

Рисунок В.1 – Архітектурна схема веб-ресурсу інтернет-магазину

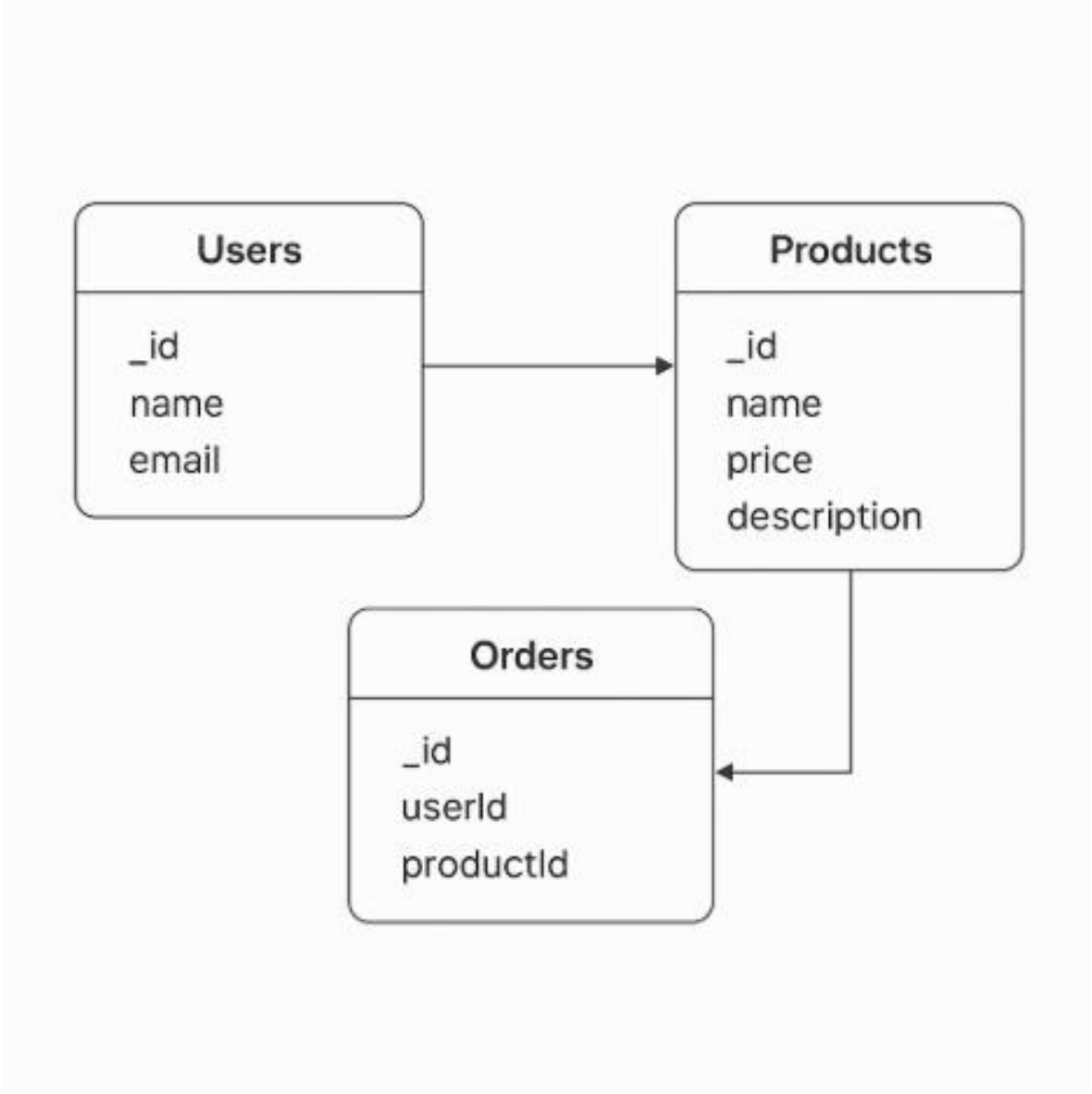

Рисунок В.2 – ER-діаграма бази даних веб-ресурсу

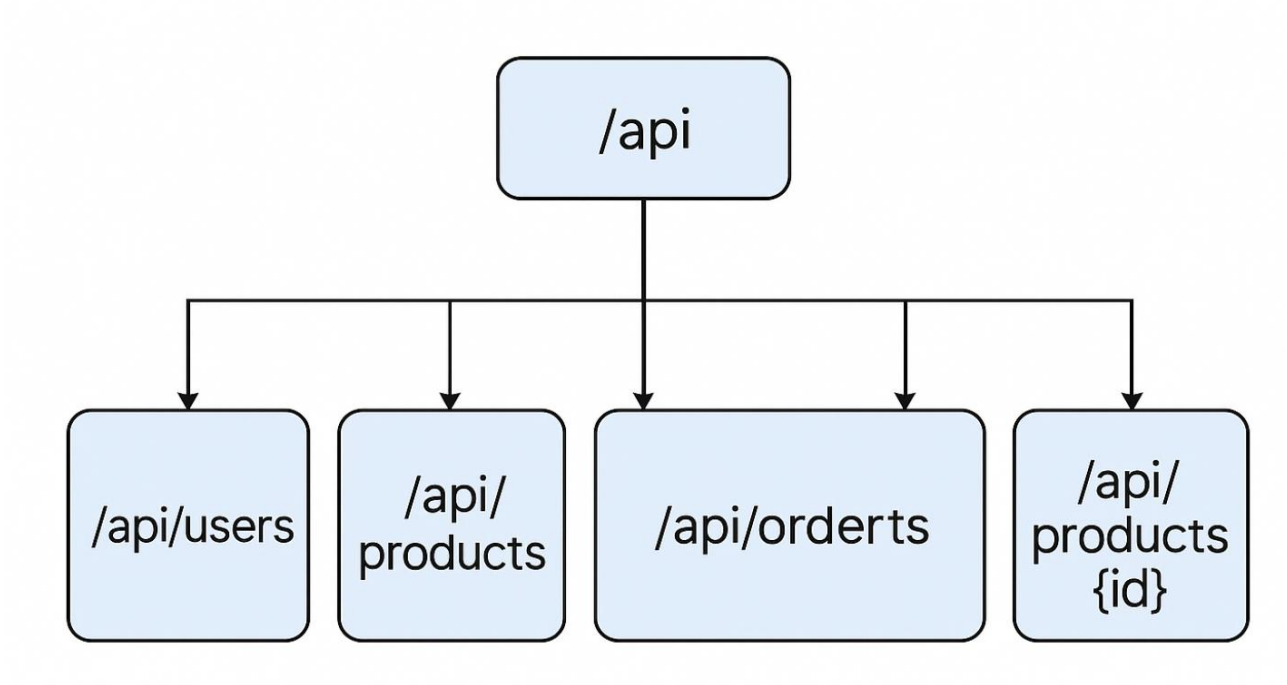

Рисунок В.3 – Схема маршрутів API веб-ресурсу

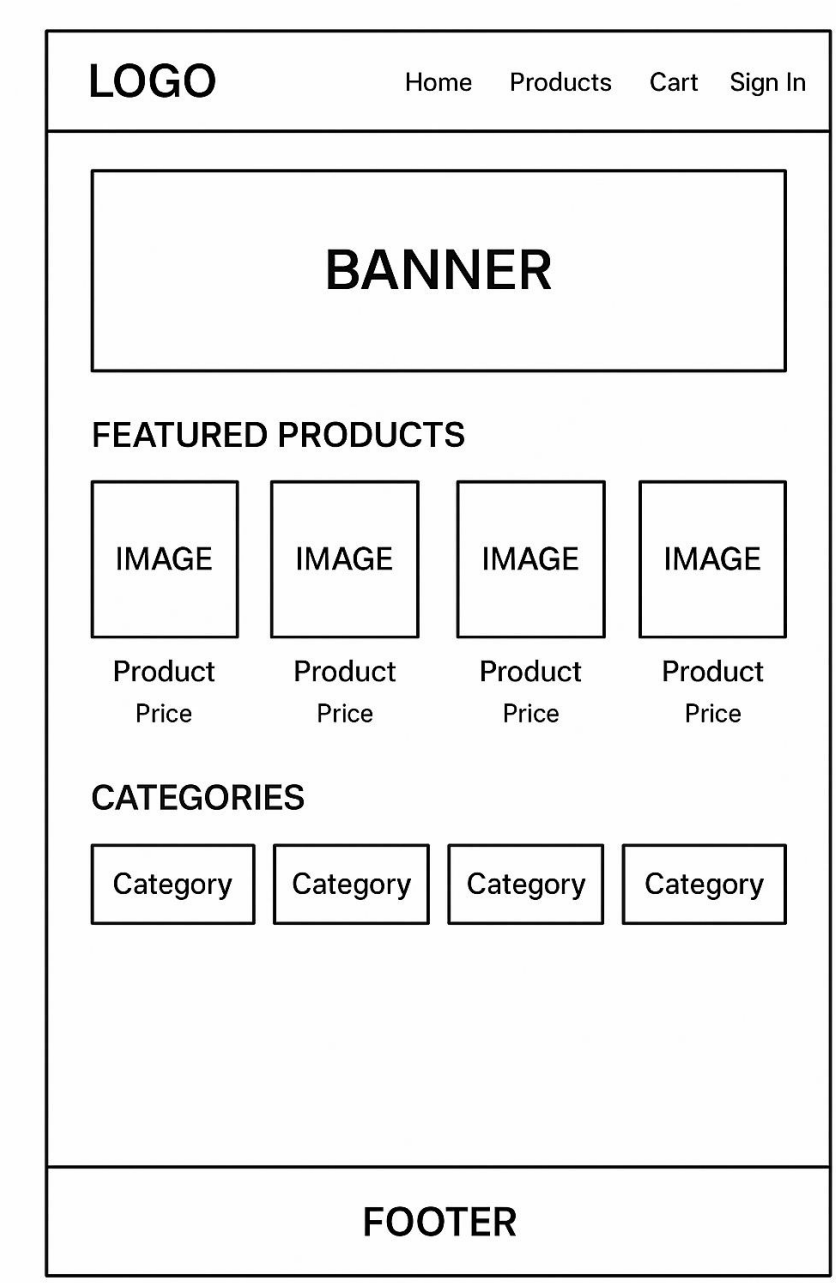

Рисунок В.4 – Макет головної сторінки веб-ресурсу інтернет-магазину

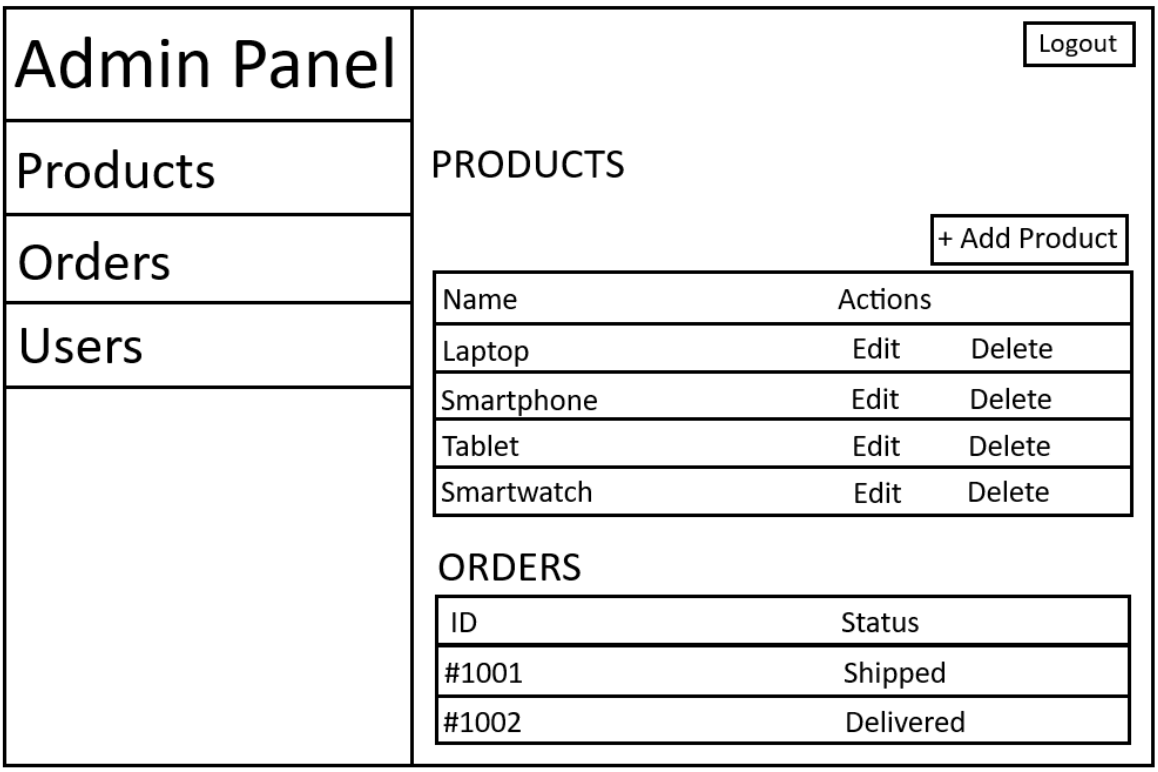

Рисунок В.5 – Інтерфейс адміністративної панелі для керування товарами та замовленнями

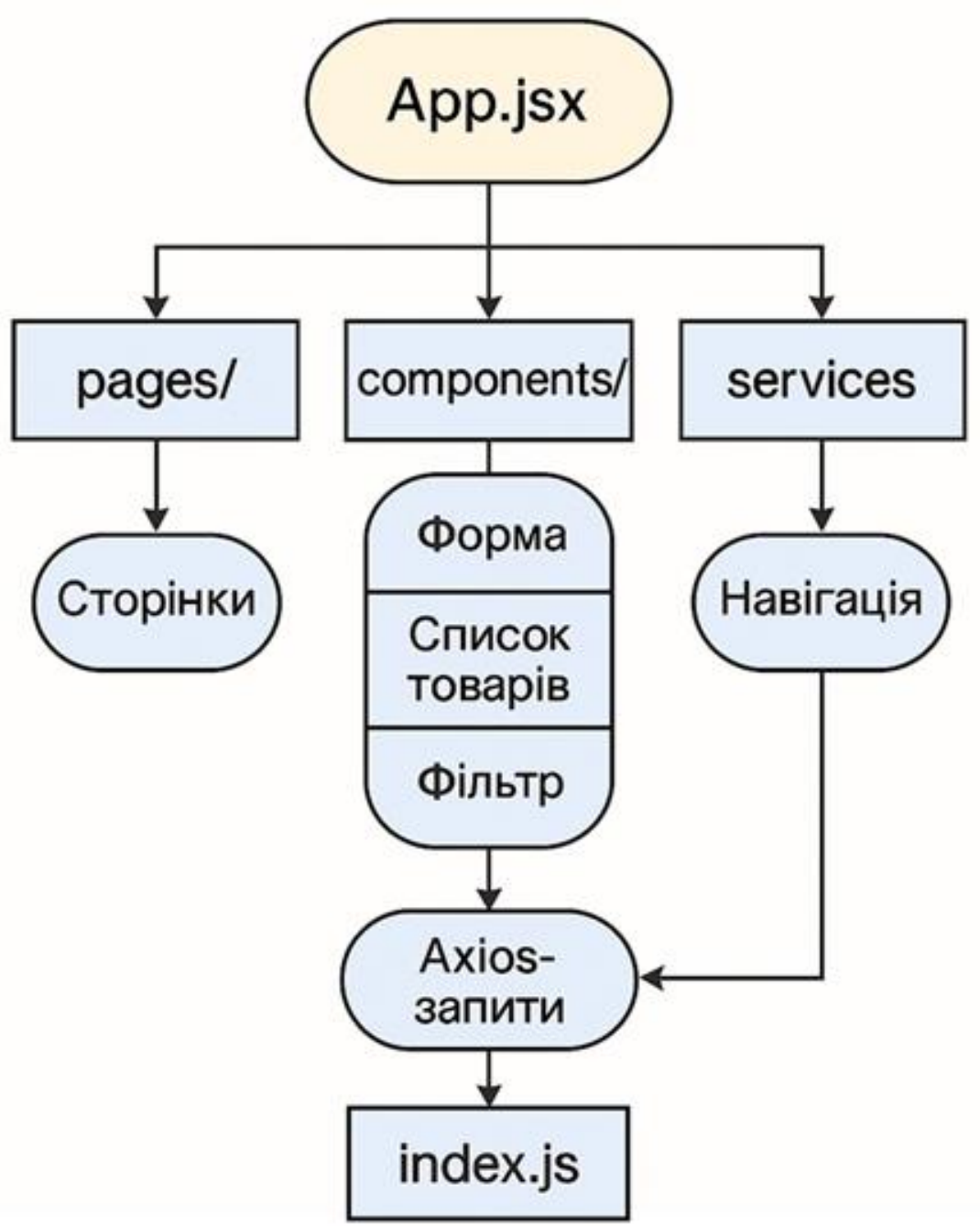

Рисунок В.6 – Структура клієнтської частини

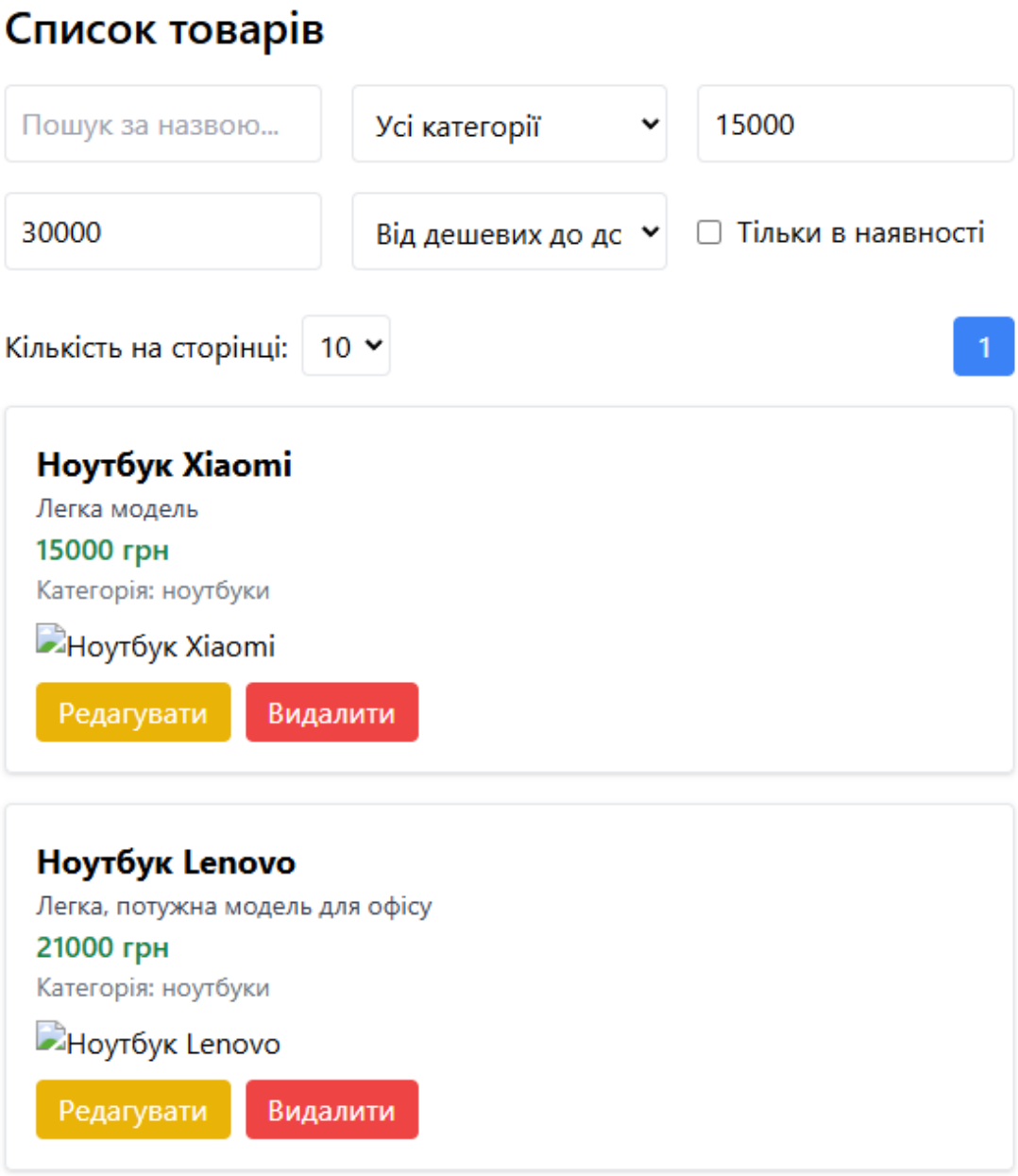

Рисунок В.7 – Відображення списку товарів з фільтрами та кнопками для перегляду/редагування/видалення

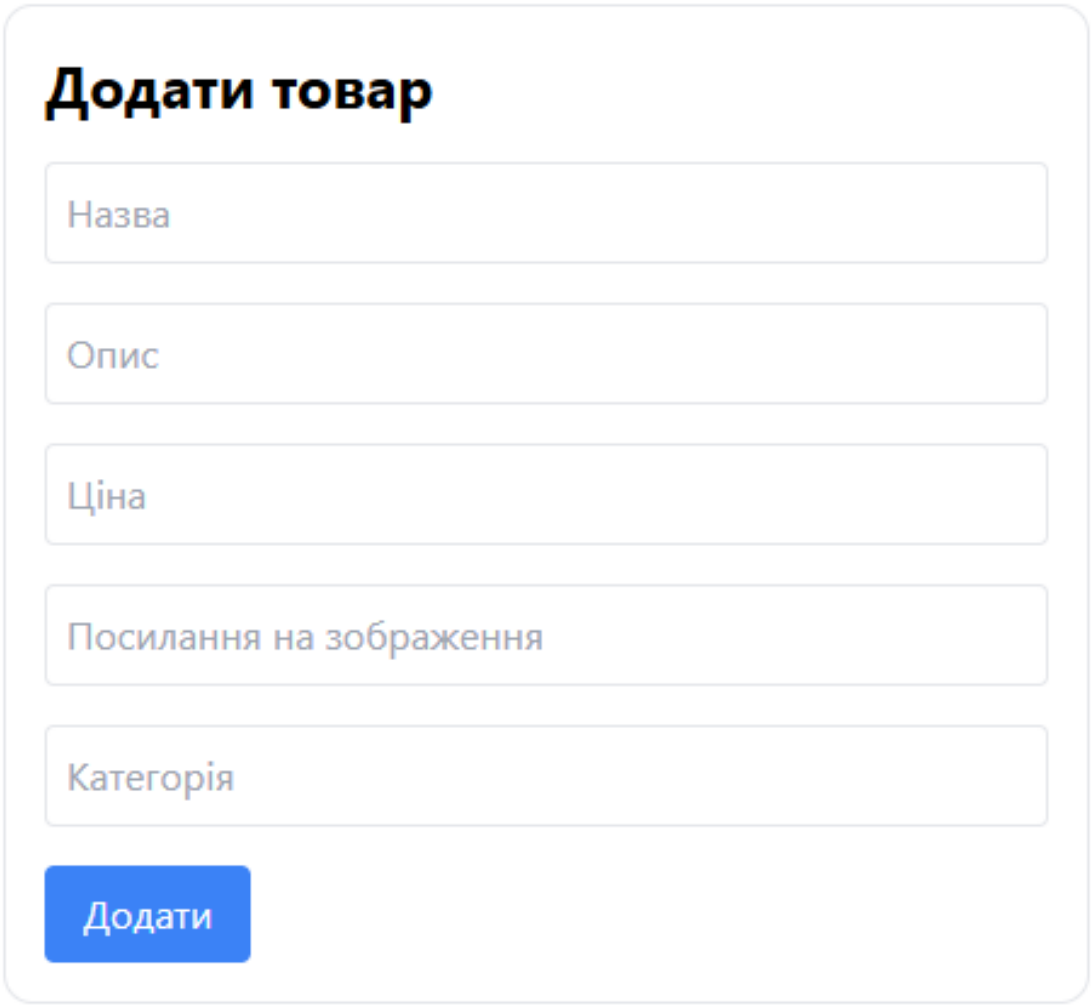

Рисунок В.8 – Форма додавання нового товару

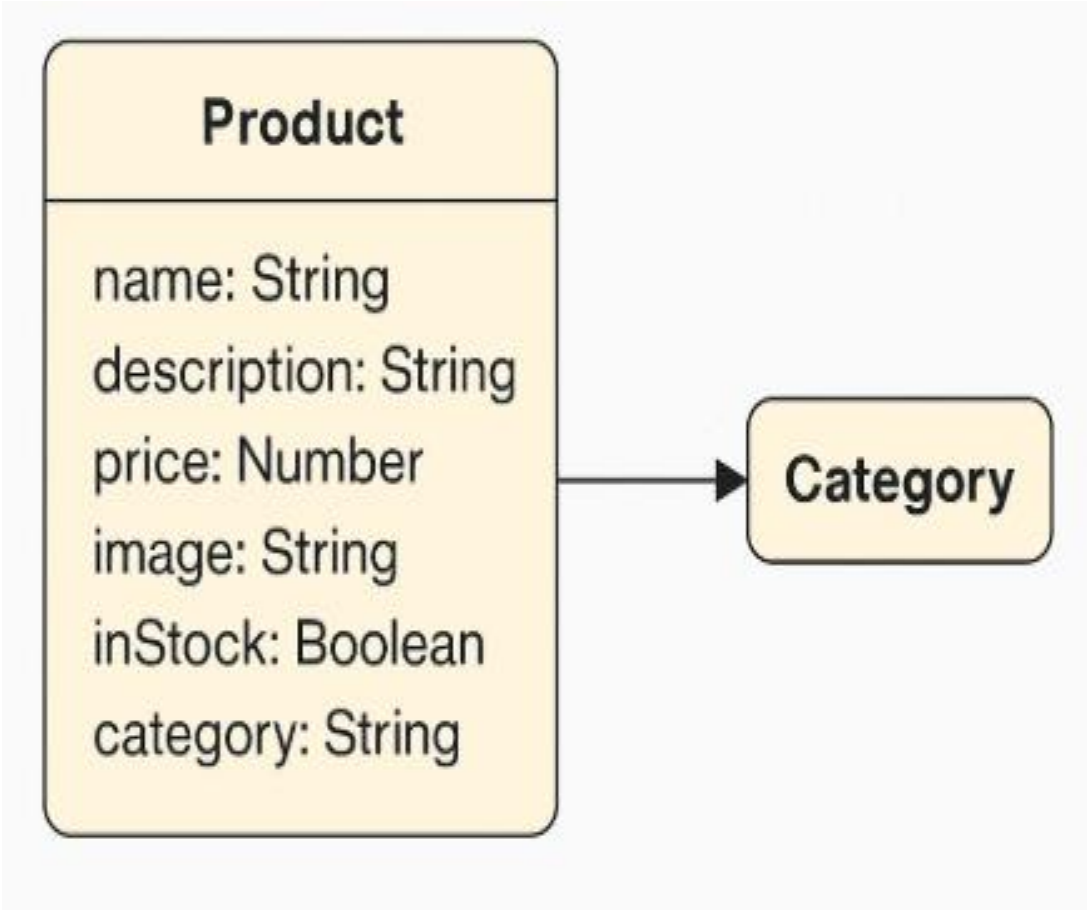

Рисунок В.9 – Схема Mongoose-моделі Product (ER-діаграма)

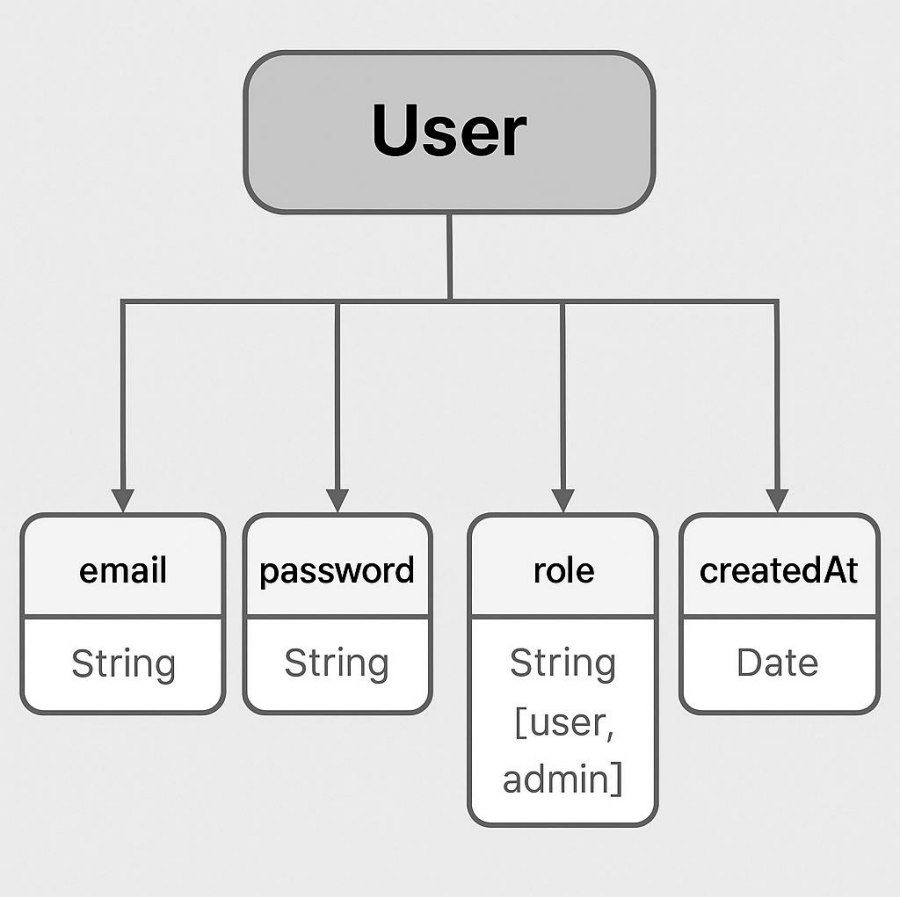

Рисунок В.10 – Структура моделі User

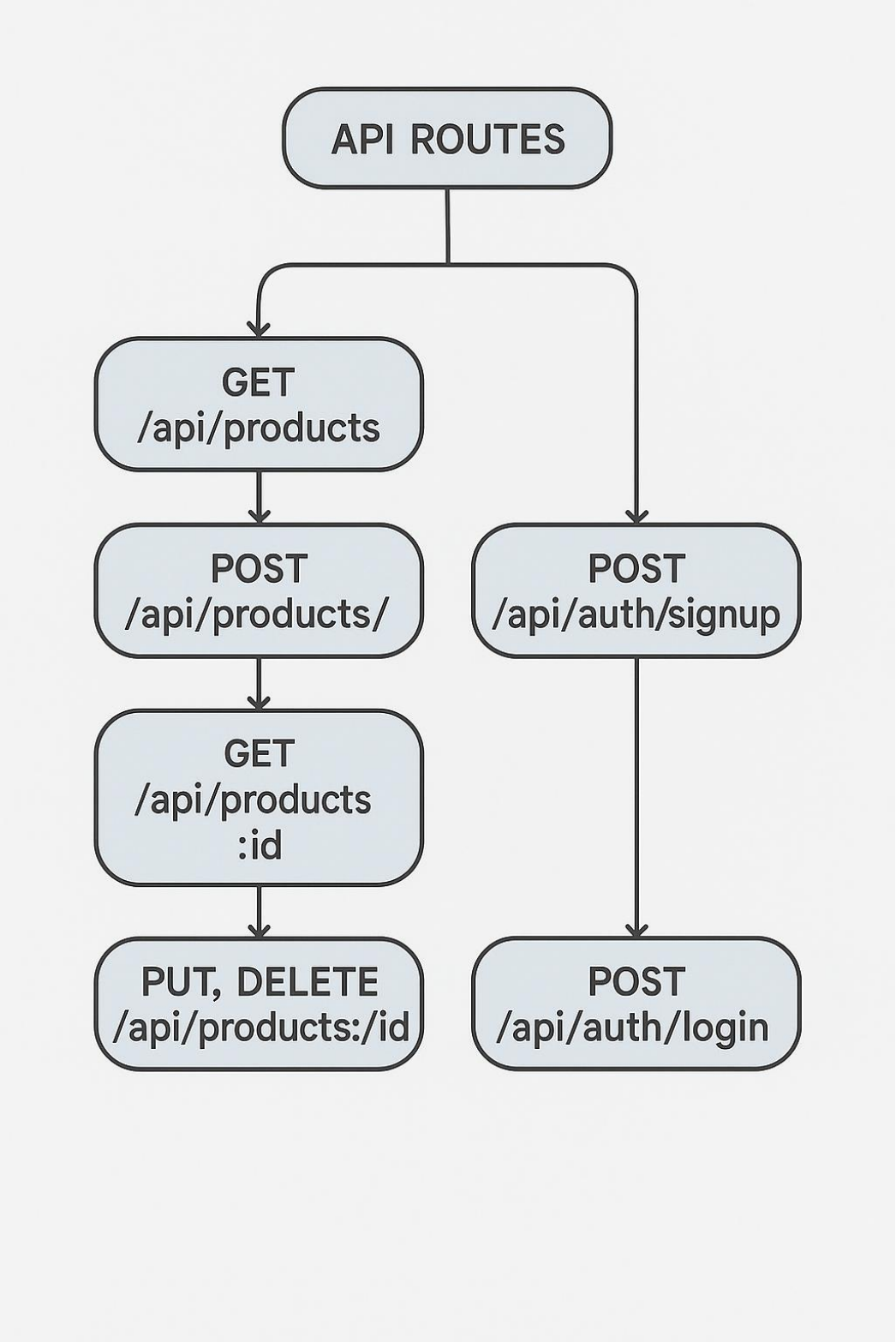

Рисунок В.11 – Схема роботи маршрутів API

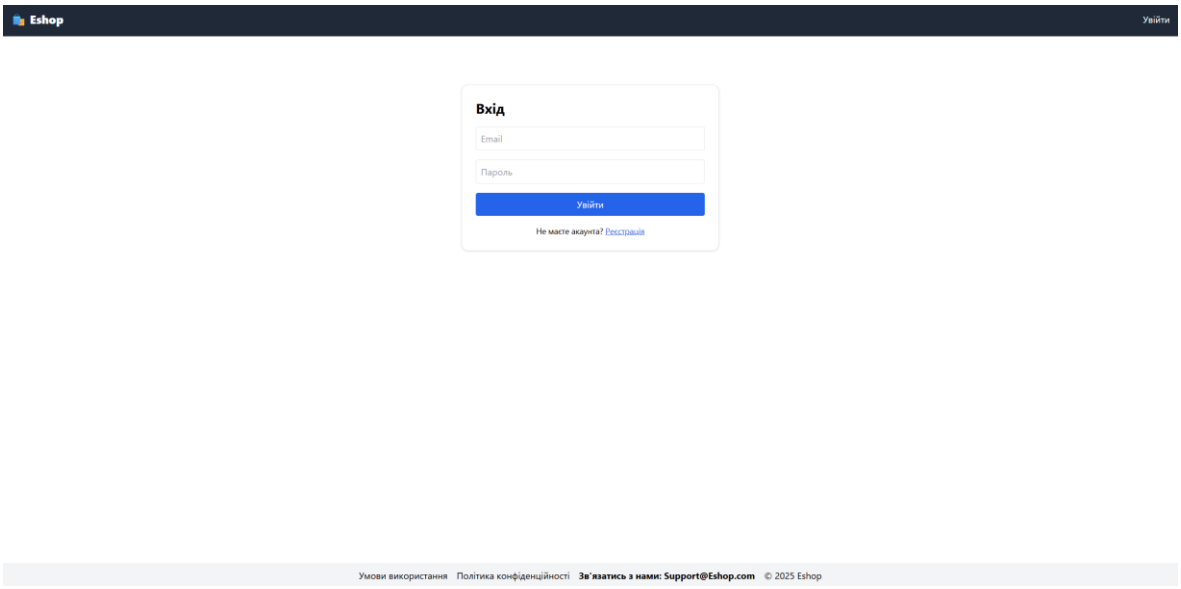

Рисунок В.12 – Головна сторінка веб-ресурсу для неавторизованого користувача

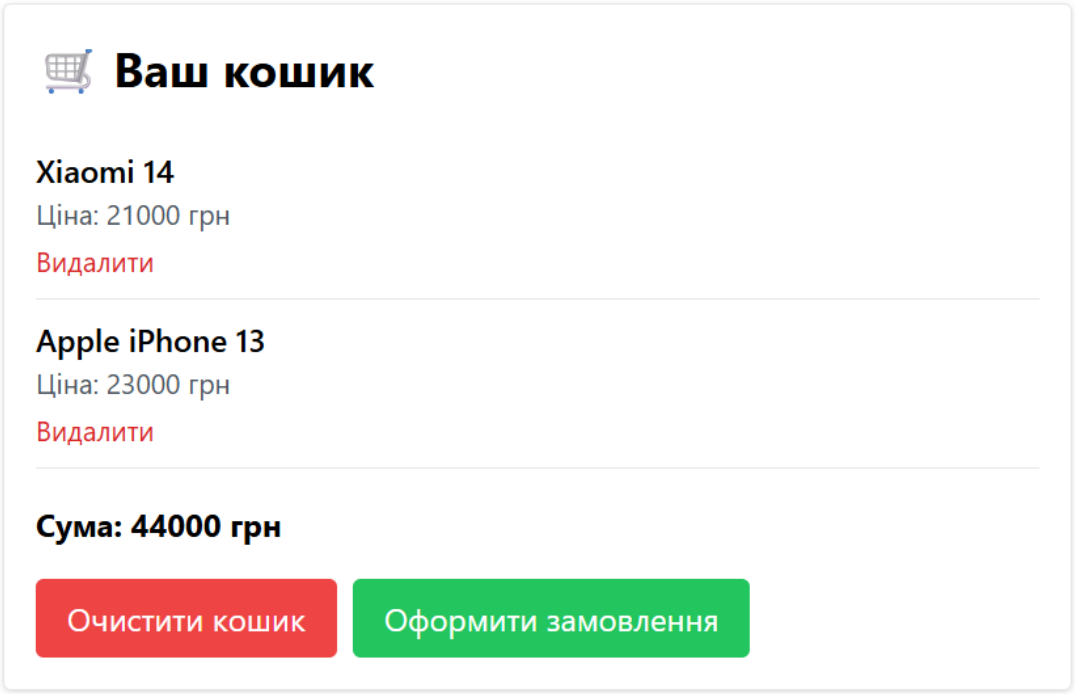

Рисунок В.13 – Кошик користувача з доданими товарами

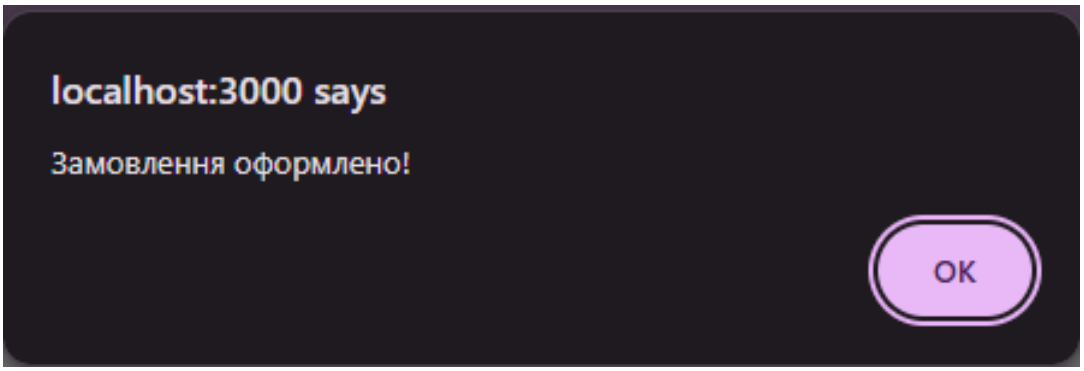

Рисунок В.14 – Повідомлення про успішне оформлення замовлення

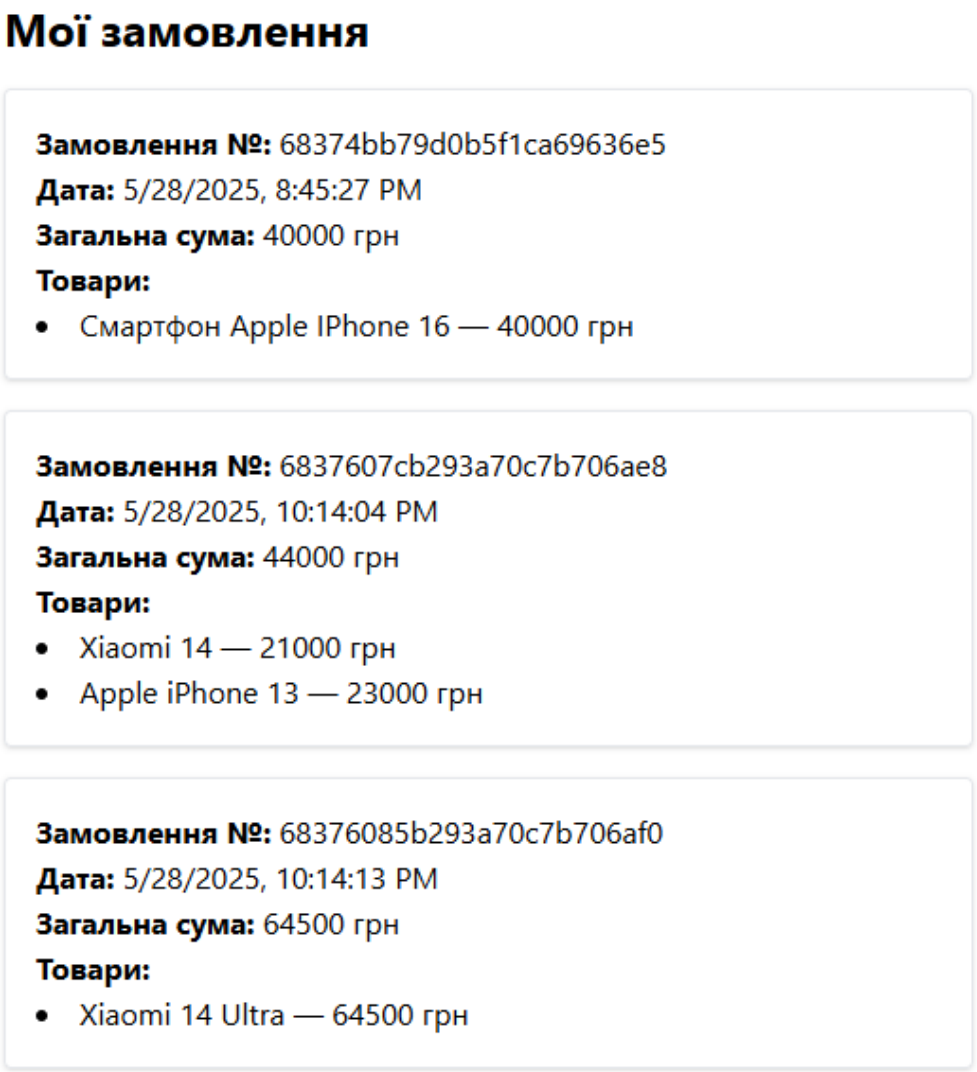

Рисунок В.15 – Меню замовлень користувача

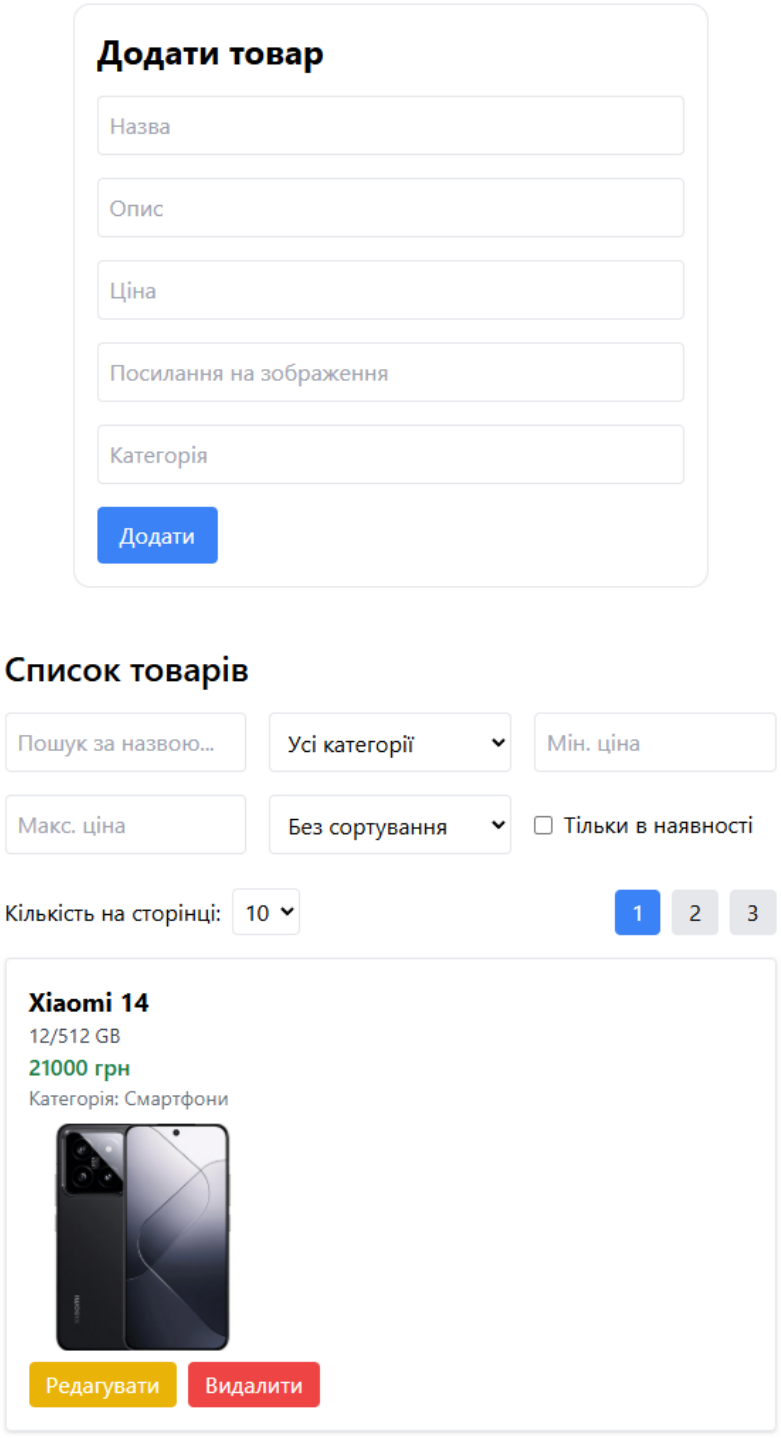

Рисунок В.16 – Адміністративна панель управління товарами

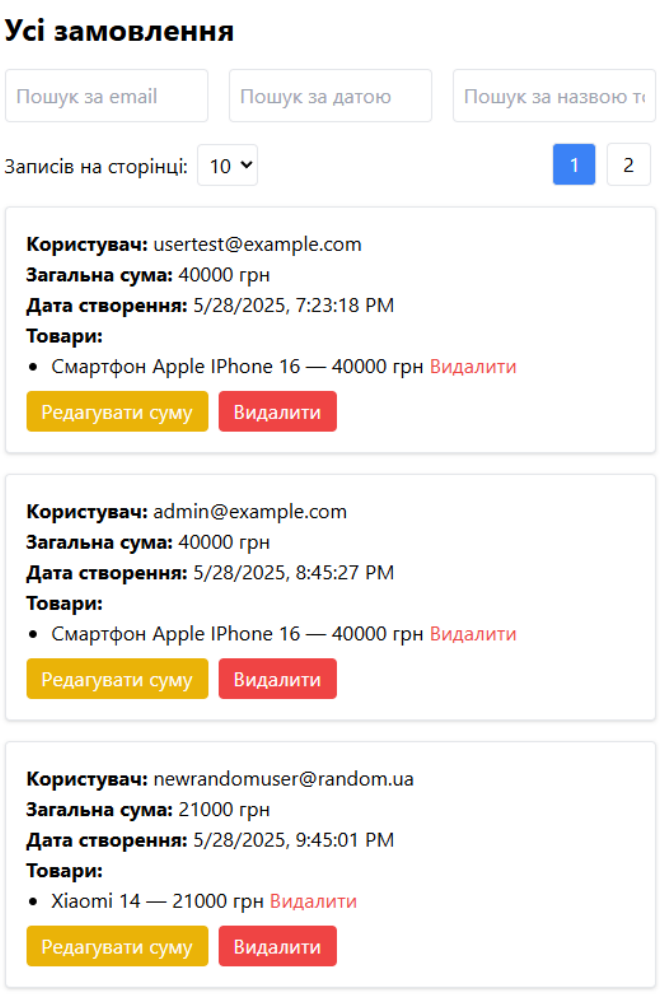

Рисунок В.17 – Перегляд усіх замовлень в адміністративній панелі

ДОДАТОК Г (ДОВІДНИКОВИЙ)

Інструкція користувача

Ця інструкція описує кроки, які необхідно виконати для запуску проєкту веб-ресурсу інтернет-магазину електроніки та відкриття його у браузері.

Перш за все необхідно встановити наступні компоненти:

- Node.js (версія 18 або вище) з офіційного сайту Node.js;
- MongoDB Community Server з офіційного сайту MongoDB
- Опціонально: Visual Studio Code як середовище розробки з офіційного сайту Visual Studio Code

У кореневій директорії створіть файл .env для бекенду (server/.env) і вкажіть у ньому змінні оточення:

PORT=5000

MONGO_URI=mongodb://localhost:27017/electroshop

JWT_SECRET=your_jwt_secret

Перейдіть у папку бекенду (/server) та встановіть залежності:

cd server

npm install

Аналогічно для клієнтської частини (/client):

cd client

npm install

Тепер для запуску серверної частини використовується (бекенд):

cd server

npm run dev

Для клієнтської частини (фронтенд):

cd client

npm start

Після запуску веб-ресурс буде доступний за адресою: http://localhost:3000